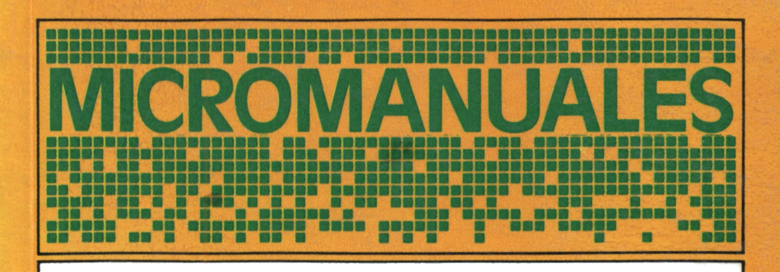


Introducción a los Sistemas Operativos

Blackburn & Taylor

ANAYA
MULTIMEDIA

INTRODUCCION A LOS SISTEMAS OPERATIVOS

INTRODUCCION A LOS SISTEMAS OPERATIVOS

Lawrence Blackburn
Marcus Taylor



MICROMANUALES

Titulo de la obra original :
INTRODUCTION TO OPERATING SYSTEMS

Traducción de: Fernando García Fernández
Diseño de colección: Narcís Fernández

Villafranca, 22. 28028 Madrid
Depósito legal: M-17816-1986
ISBN: 84-7614-092-4
Printed in Spain
Imprime: LIBROGRAF, S.A.
Molina Seca, 13. FUENLABRADA (Madrid)

Indice

Cómo usar este manual

Hemos escrito este manual para ayudar a aquellos que necesitan una introducción general al tema de los sistemas operativos o para los que quieren generalizar los conocimientos adquiridos con un sistema operativo específico. Puede ser un excelente resumen para los estudiantes de Informática que tengan en su plan de estudios una asignatura que estudie los sistemas operativos. También puede servir como libro introductorio para otros manuales y guías de bolsillo dedicadas a sistemas operativos específicos: UNIX, CP/M y MS-DOS, por ejemplo. Al contrario que otros libros, que se usan mejor sentados delante del ordenador, éste se puede usar como manual de referencia, ya que ofrece descripciones concisas de conceptos y líneas maestras sobre las que se pueden desarrollar los conocimientos al trabajar en un entorno de desarrollo de sistemas.

El desarrollo de un sistema operativo puede tardar varios años y en cada caso específico tendrá un tamaño y configuración diferente que le permitirá adaptarse mejor a un ordenador y/o a una tarea específica. Este manual ha omitido muchos detalles al discutir determinados elementos, pero esperamos

haber conseguido nuestra misión de describir un sistema operativo de la complejidad del UNIX, típico de miniordenadores. Las palabras escritas en *cursiva* pueden indicar dos cosas: o que se quiere hacer destacar ese tema o que en páginas posteriores se explica más detalladamente dicha palabra. Se ha dedicado el último capítulo a comparar las características de los principales sistemas operativos para microordenadores que existen actualmente.

La primera ola de microordenadores de 16 bits ha logrado que una gran cantidad de sistemas operativos que solían estar reservados a sistemas mucho más caros estén disponibles para mucha gente. Este libro, así como otros de la serie, está destinado en definitiva a ayudarle a familiarizarse con los conceptos generales referentes a los sistemas operativos y, por tanto, a servir de guía de referencia en su cartera o al lado de su ordenador.

1

Introducción

La diferencia entre la mayoría de las máquinas y los ordenadores es que, debido a la capacidad que tienen estos últimos de ser programados, pueden realizar una amplia gama de trabajos; son dispositivos de *propósito general.*

Esta generalidad lleva a su dificultad de uso. Como un primer paso para comprender los sistemas operativos es necesario asimilar los propósitos esenciales para los que existen los sistemas operativos. Esta sección muestra y explica dichos fundamentos.

Un propósito de un sistema operativo es esconder al usuario las numerosas partes componentes de modo que al usarlo sea sencillo. Sin embargo, para comprender cómo responde a las instrucciones que se le dan, es necesario describir detalladamente el funcionamiento de sus partes y examinar cómo funcionan juntas para conseguir el objetivo de manejar los recursos del ordenador. En las máquinas que tienen definida claramente su función, como por ejemplo una máquina de escribir, es relativamente sencillo identificar qué bloques (o mecanismos) se moverán cuando se pulsa una tecla y se tiene por resultado un carácter impreso en una hoja de papel. Por esta razón hay un número considerable de máquinas de escribir altamente competitivas a la

venta. Pero para un ordenador de propósito general, que no es un dispositivo con una sola finalidad, es más difícil identificar qué mecanismos serán útiles normalmente y, además, cómo se deben relacionar unos mecanismos con otros. La misión de un sistema operativo es permitir que la máquina sea usada como un ordenador de propósito general. Le añade al ordenador un conjunto de funciones útiles análogas al mecanismo que traslada una pulsación de una tecla en un carácter impreso.

ORDENADORES DE PROPOSITO GENERAL

Aunque existe un gran número de modelos de ordenadores de propósito general (desde micros hasta *mainframes*), las presiones del mercado han provocado una estandarización en la forma de funcionamiento de los bloques principales de un ordenador. En esencia el ordenador es una máquina que ejecuta una secuencia de instrucciones que tiene almacenadas electrónicamente. Normalmente el repertorio de instrucciones incluye unos cuantos cientos distintos y la memoria de la máquina permite almacenar decenas de miles simultáneamente. Hoy en día se pueden adquirir en muchos comercios, incluso en los grandes almacenes, ordenadores de propósito general con unas especificaciones similares de CPU *(Central Processing Unit* o Unidad Central de Pro - ceso) y de almacenamiento primario (memoria) a las que acabamos de indicar. Pero para que un ordenador pueda sacar todo el rendimiento posible de estos dos componentes, y manejar complejas aplicaciones de negocios y de proceso de datos, hace falta que exista un sistema operativo diseñado correctamente que sea capaz de controlar todas las funciones. Este pequeño

texto intenta describir los principios generales de diseño empleados en la construcción de un sistema operativo capaz de manejar correctamente todos los componentes del equipo.

El almacenamiento secundario, normalmente cintas y discos, compensa las deficiencias que presenta a nivel de capacidad el almacenamiento primario, pero es considerablemente más lento al recuperar los datos si es que éstos no se encuentran ya en memoria.

Los archivos contenidos en almacenamiento secundario reciben un nombre único por razones de identificación, ya que un ordenador típico de gestión o de investigación puede contener cientos de archivos simultáneamente en el almacenamiento secundario. Esto puede crear un problema (o una espera) si el ordenador no es capaz de localizar un archivo aleatorio en un momento dado. Para evitarlo se reserva una zona especial del almacenamiento secundario en la que se guarda una lista de los nombres de los archivos y de las posiciones que ocupan en dicho almacenamiento secundario, y se crean procedimientos especiales que sean capaces de procesar estas listas tan rápidamente como sea posible (véase capítulo 7).

COMUNICACIONES

El usuario/operador necesita comunicarse con el ordenador para poder realizar tareas. Para esta función normalmente se utiliza un terminal de ordenador. Por este terminal, que está compuesto por un teclado similar al de una máquina de escribir y por una pantalla similar a la de televisión, pasan las órdenes enviadas por el usuario al ordenador y los mensajes de respuesta de éste. Actualmente se encuentran otros sistemas más fáciles de manejar bajo

desarrollo, pero el sistema compuesto por teclado-pantalla seguirá en activo durante bastante tiempo. La conexión del terminal al ordenador puede hacerse de dos formas: directamente, con un cable que los una, o indirectamente por medio de un *modem* conectado a la red telefónica pública. La primera opción es la más rentable, pero la última es la única posible cuando ambos dispositivos se encuentran muy alejados.

Casi todos los ordenadores de propósito general tienen las características técnicas descritas. Pero sin programas se encuentran "desnudos". Una máquina que se encuentre en este estado funciona colocando una serie de interruptores con una configuración determinada que le indica cuál es la siguiente operación que debe realizar. Los primeros ordenadores que aparecieron funcionaban de este modo y, por tanto, resultaban muy difíciles de manejar. Para simplificar esta tarea y que no hiciese falta constantemente la presencia de un técnico especializado se inventaron los sistemas operativos. Los técnicos siguen existiendo, pero su trabajo ya no implica una relación tan estrecha con la máquina. Hoy en día todos los ordenadores de propósito general utilizan sistemas operativos para evitar los problemas mencionados anteriormente. Por tanto, el propósito de un sistema operativo es aumentar la efectividad de un ordenador de propósito general.

Hemos definido un sistema operativo como un programa que maneja los elementos de un ordenador de propósito general. Si queremos describirlo haciendo referencia a los componentes del *hardware* y a sus relaciones, por ejemplo enviar información a los terminales, podemos decir que se encarga de manejar el almacenamiento primario y el secundario (un disco, por ejemplo) y de transferir información del almacenamiento secundario al primario. Si el ordenador controla a varios usuarios a la vez, es el sistema operativo el que se encarga de manejar las demandas

que se realizan de los dispositivos a su cargo y decide cómo ordenar las tareas para optimizar la utilización del sistema.

REQUERIMIENTOS DEL SISTEMA OPERATIVO

La complejidad de un sistema operativo depende de la complejidad de la máquina sobre la que esté funcionando. Si ésta es muy compleja (muchos dispositivos distintos conectados, varios usuarios, etcétera), el sistema operativo será complejo, ya que necesitará más programas para manejar todos estos dispositivos y todas las tareas. Por otra parte, los ordenadores sencillos tienen sistemas operativos sencillos. Un tipo incluido en la categoría de los sencillos es un monitor; este programa tiene un conjunto muy limitado de instrucciones que sólo son capaces de realizar las tareas más básicas del sistema. A veces la complejidad de un sistema operativo proviene de la necesidad de manejar más de un usuario a la vez. Para poder hacer esto se han desarrollado técnicas de tiempo compartido que permiten compartir los distintos elementos componentes del ordenador entre los distintos usuarios sin que surjan conflictos entre éstos. Otros métodos para cambiar de un trabajo a otro han evolucionado hasta producir un conjunto de técnicas denominadas *multiprogramación.*

Los ordenadores de un solo usuario (por ejemplo, los ordenadores personales) se diseñan para ejecutar un programa cada vez; este tipo de máquinas no necesitan multiprogramación. Los grandes sistemas de ordenadores que son capaces de realizar varios millones de operaciones por segundo necesitan técnicas de tiempo compartido que permitan distribuir

el tiempo libre de la CPU entre una tarea y otra; cuando al ejecutar un programa se produce una espera, el sistema debe saltar a otra rutina y cuando ésta se detenga por alguna razón natural, entonces el control pasa a otra rutina. De esta forma la CPU está ocupada continuamente atendiendo a varios trabajos a la vez.

Si tenemos en cuenta que existe una penalización de tiempo cada vez que se pasa información del almacenamiento secundario al primario (¡un millón de instrucciones de CPU aproximadamente!) y que todas las comunicaciones del ordenador las maneja el sistema operativo, entonces resulta evidente la conveniencia de que todas las funciones del sistema operativo que son usadas frecuentemente se encuentren permanentemente en el almacenamiento primario. Esto reduce el número de veces que se accede al almacenamiento secundario por petición del sistema operativo durante su funcionamiento. La parte del sistema operativo que reside permanentemente en memoria se llama *núcleo* o *kernel*. El tamaño del núcleo se determina en función de muchos factores, por ejemplo: número de usuarios, memoria disponible, memoria reservada para *buffer* y tiempos de respuesta aceptables. En general, cuanto más grande y compleja es la máquina, más memoria necesita el núcleo.

FLEXIBILIDAD

El sistema operativo es uno de muchos programas (aunque muy importante) que se tiene que cargar en el almacenamiento primario antes de ejecutarse. El

programa se ejecuta secuencialmente por la CPU instrucción a instrucción según el orden en que están dispuestas en memoria hasta que llega una que rompe el flujo secuencial normal. Los programas son importantes porque constituyen el único interfaz existente entre el usuario y la máquina desnuda. En un funcionamiento típico el usuario proporciona cierta información (normalmente poca) y el ordenador devuelve otra como respuesta. En un sistema bien diseñado la máquina gasta más tiempo trabajando para el usuario que el usuario trabajando para la máquina. Además los programas se deben diseñar para que tengan el uso más amplio y versátil posible, al igual que el ingeniero diseña la máquina desnuda lo más genéricamente posible. Por ejemplo, un programa que cambia el nombre a un archivo debe diseñarse lo más genéricamente posible de modo que pueda cambiar de nombre a cualquier archivo y no a uno solo. En este caso el usuario debe proporcionar dos nombres —uno el nombre actual, el otro el nuevo nombre— y el programa hará el resto. Un programa de este tipo se incluye en el sistema operativo, incluso formando parte del núcleo como *orden,* dado que resulta evidente que es una función importante.

Los programas se pueden dividir en dos categorías importantes: Utilidades y aplicaciones. Los de utilidades normalmente ejecutan acciones generales mientras que los de aplicaciones se diseñan con un propósito específico. Ejemplos de programas de utilidad son: un programa que da la hora, o uno que lista los archivos en disco. Los de aplicación serán, por ejemplo, los que realicen la nómina o la contabilidad. Las utilidades normalmente vienen con el sistema operativo, mientras que los programas de aplicación se suelen comprar y/o desarrollar posteriormente.

Los siguientes capítulos examinarán en detalle cada uno de los elementos que hemos explicado aquí y detallarán los elementos más importantes de cada área

global, dando importancia a los aspectos de manejo de los diversos elementos del sistema, ya que se considera que el sistema operativo es una herramienta para manejar los diversos componentes.

2

Las funciones y el núcleo

Después de más de treinta años de experiencia con ordenadores, parecería razonable la existencia de un concepto común de lo que es un sistema operativo. Pero esto no es así. En la literatura especializada se sigue discutiendo cuál debe ser la lista de requerimientos funcionales de un sistema operativo y qué parte de cada función debe admitirse como componente del sistema operativo. A continuación se da una lista de los componentes y tareas sobre los que hay acuerdo de que deben ser manejados por el sistema operativo:

- El procesador (ejecución interactiva de programas, manejo de interrupciones y acceso a los recursos almacenados).
- La memoria (asignación de este recurso bajo demanda de los programas de los usuarios cuando se les activa).
- Los dispositivos de entrada y salida (inicialización de los dispositivos, manejo de peticiones y control de procesos).

- Inicialización de programas y comunicaciones entre procesos.
- Los datos (apertura, lectura, escritura y cierre de archivos en los dispositivos secundarios).

La existencia de unas funciones básicas capaces de realizar estas tareas implica que el usuario dispone de un conjunto de recursos lógicos que son más fáciles de usar que el *hardware* sobre el que operan. Esto también implica que varios usuarios pueden compartir dichos recursos bajo control del sistema. Dentro de estas definiciones generales de las funciones de un sistema operativo existen tantas variaciones, que cada uno de los existentes parece totalmente distinto a los otros debido al modo en que realiza las diversas funciones. Esta sección intenta concentrarse en los aspectos funcionales de los sistemas operativos que:

- Definen un entorno para la creación y ejecución de programas.
- Crean un conjunto de mecanismos para acceder a los recursos.
- Crean un interfaz para realizar determinadas operaciones.
- Manejan los recursos del sistema.

Para conseguir que un sistema operativo haga esto, hay que reservar parte de la zona de almacenamiento principal para almacenar en ella el *núcleo*. Es decir, el conjunto de programas de control que deben residir en la memoria principal para mantener un control continuo de los recursos.

FUNCIONES DEL SISTEMA OPERATIVO

Entornos para la creación y ejecución de programas

Toda la actividad de un ordenador viene determinada por la ejecución de programas. La realización de estos programas requiere diseño, desarrollo y comprobación. Para que la producción de *software* sea rentable, hace falta un *entorno* que proporcione soporte para la realización de estas tareas. En el capítulo 9 se da una explicación más detallada de este entorno; no obstante, es conveniente saber lo que un programador desea tener en un sistema operativo:

- Describir el problema o solución en un lenguaje conveniente.
- Recibir ayuda durante la comprobación del programa.
- Desarrollar partes funcionales de un programa independientemente de otras (módulos).
- Integrar varios módulos entre sí cuando sea necesario.

Estos requerimientos los satisfacen respectivamente los compiladores, *debuggers, linkers* e interfaces del sistema operativo. La función de un *linker* es combinar programas (módulos) que se desarrollan por separado en un solo programa que se tratará como una sola unidad. La salida generada por un *linker* se carga en memoria con un *loader* y recibe recursos por medio de un programa asignador de recursos. Una vez cargado, el programa se activa y entra en un entorno de ejecución. Cuando el programa está funcionando sólo ve aquella parte del sistema total a

la que le permite acceder el entorno de ejecución. Normalmente éstas serán las utilidades y servicios que se muestran en la figura 1. Aparte de éstos, existen muchos otros servicios que no son visibles por el programa que se está ejecutando, como son instrucciones privilegiadas e instrucciones de protección de memoria, entre otras. En realidad lo que hace el entorno de ejecución es ampliar las funciones de la máquina para que los compiladores (y los programadores de lenguaje máquina) puedan crear un programa objeto de más alto nivel.

PROGRAMA DE APLICACION	OPEN GET PUT READ WRITE	MEMORIA PROTEGIDA MEMORIA INSTRUCCIONES	OPERACIONES PRIVILEGIADAS (No vistas)

Figura 1. *Entorno de ejecución.*

En los primeros sistemas operativos monousuario el compilador que generaba el código ejecutable no sólo tenía que escribir y leer los datos de los *buffers*, sino que también tenía que inicializar las operaciones de entrada y salida de los distintos dispositivos para vaciar o llenar estos *buffers* cuando se llenaban o se empleaban. Desde la aparición de los sistemas operativos multiusuario capaces de ejecutar dos o más programas simultáneamente, fue necesario introducir el concepto de estado de control junto a la "máquina extendida". El *Control de estado* se desarrolló por la necesidad de establecer un mecanismo de coordinación entre varios programas que no eran conscientes de su existencia mutua. El estado de control prohibe la aparición y ejecución de ciertas tareas que pueden perturbar la integridad de los programas concurrentes (véase capítulo 6). Por este

motivo, la idea actual de entornos de ejecución contiene la idea de coordinación y control así como la de interfaces ampliados. Los compiladores actuales no generan todas las instrucciones de entrada y salida necesarias para el programa; dichas instrucciones (entre otras) son privilegiadas y para poder ejecutarse el ordenador debe encontrarse en un estado especial de control. Si el programa desea ejecutar dichas funciones debe realizar una llamada al sistema supervisor que pone al procesador en estado de control y ejecuta una rutina especial de entrada y salida contenida en el sistema operativo. Para hacer esto la llamada al supervisor genera una interrupción que es detectada y servida por el sistema operativo. Una *interrupción* es un suceso que hace que la CPU deje de procesar instrucciones de un usuario para que pueda controlar o atender una actividad más urgente del sistema.

Entorno para acceso del usuario

Los recursos de un ordenador existen para ser usados y son accesibles al usuario o a un programa determinado a través del sistema operativo. El grado de accesibilidad que exista vendrá determinado por la comunicación existente entre el usuario y los recursos del ordenador. Este sistema de comunicación con el usuario permite que éste realice el control general de una tarea. Para realizar dicho control de la ejecución de un programa existe un *lenguaje de control,* denominado así en contraposición al lenguaje de programación que se usa para describir la solución contenida en el programa.

El tipo de acceso que permiten los diversos sistemas operativos refleja las definiciones que realiza acerca de las diferentes clases de usuarios, por ejemplo, operadores, programadores y programadores del

sistema. Algunos sistemas operativos tratan a todos los terminales que se encuentren en línea como si fuesen consolas de operador, de modo que las funciones de control y de funcionamiento se mezclan. Otros sistemas determinan el tipo de usuario y en función de este tipo le proporcionan un conjunto de órdenes.

El lenguaje de control encargado de controlar los recursos del sistema y el acceso del usuario ha ido evolucionando a lo largo de los años para adaptarse a los avances producidos en el *hardware* y las nuevas habilidades de los sistemas de comunicación para adaptarse a unidades remotas. El resultado es la gran variedad de tipos de acceso disponibles para manejar el sistema. La siguiente lista da un resumen de los más importantes:

1. *Acceso por petición de hora.*—El usuario pide el uso del sistema con anticipación y cuando puede acceder tiene total control sobre él. El funcionamiento es similar al alquiler de una pista de tenis. Fue uno de los primeros sistemas que se desarrolló para acceder a un sistema y se sigue usando cuando se deben ejecutar tareas fundamentales de mantenimiento del sistema.

2. *Acceso local por lotes.*—El trabajo del usuario se añade a una cola de trabajos que se prepara (fuera del ordenador) para introducirlos todos simultáneamente en un momento dado. Una vez entregados todos, el encargado del sistema forma una lista interna de peticiones de trabajo dando instrucciones en el lenguaje de control que describen las prioridades y las necesidades de recursos. Este método lo utilizan muchos sistemas de tercera generación.

3. *Acceso por introducción remota de trabajos (RJE-REMOTE JOB ENTRY)* .—El usuario activa un trabajo sin necesidad de que exista un operador, sino que es aceptado por un elemento del sistema operativo que lo pone en una cola de trabajos en espera. El terminal desde el cual accede el usuario puede estar distanciado geográficamente del resto del sistema. Este proceso se diferencia fundamentalmente del anterior por la eliminación del operador humano en el proceso de aceptación de trabajos y de la distribución de accesos.

4. *Acceso por lotes interactivo.*—En esta modalidad se hace un uso totalmente interactivo (por terminal de pantalla) del concepto de RJE, siendo el usuario el que programa las peticiones de trabajos. La lista de instrucciones de control se puede hacer ejecutar con una sola llamada al entorno de control de tareas del sistema operativo que se encarga de introducirla en el grupo de tareas que se están realizando. Este tipo de acceso elimina completamente la necesidad de que el usuario almacene aparte los registros que contienen las instrucciones de control que describen la tarea.

5. *Acceso al sistema por programación interactiva.*—El usuario dispone de un terminal en el que se puede programar y ejecutar la aplicación. La mayoría de los sistemas de tiempo compartido que usan BASIC son de este tipo.

6. *Acceso por red.*—La capacidad de comunicación de los ordenadores surgió debido a la necesidad de proporcionar terminales de proceso de datos allá donde hacían falta. Inmediatamente después el usuario preguntó: "¿Qué otro *hardware* está conectado a la red y se puede usar?" Actualmente el

proceso distribuido le permite al usuario acceder a otros dispositivos del sistema por medio de la red. Este área de los sistemas operativos todavía se encuentra bajo desarrollo para asegurar la integridad y seguridad de los datos comunes.

Interfaces para operaciones y manejos

El trabajo de un operador no está definido claramente en todos los sistemas. Los rápidos cambios que se producen en la relación potencia/peso tienen un efecto desestabilizador en las definiciones de trabajos. Hoy en día se produce en muchos sistemas una falta de distinción entre las funciones operativas y de *administración.* Esto también se refleja en la mezcla funcional de los lenguajes de control y del operador en los sistemas que ofrecen acceso directo al usuario.

Las operaciones aceptadas que puede realizar un operador cuando maneja un sistema incluyen:

- Cambiar la configuración del sistema (añadir y quitar dispositivos, manejo de memoria, etc.).
- Iniciar una tarea.
- Terminar o borrar una tarea.
- Averiguar el estado de una tarea.
- Cambiar la prioridad de una tarea.
- Reiniciar el sistema.

Adicionalmente el operador se comunica con el sistema por medio de la consola del sistema para intercambiar información y órdenes referentes a la terminación de trabajos, asignación de dispositivos y control de dispositivos (montado/desmontado de volúmenes).

Además de responder a los requerimientos del operador, un sistema operativo debe realizar una política de manejo del sistema para que todas las

tareas puedan terminar correctamente. Para poder realizar esta tarea existe un interfaz que acepta parámetros que condicionan el uso general de los recursos por las diversas tareas. Por ejemplo, determina cuántos procesos concurrentes pueden existir, cuánta memoria se reserva para programar dispositivos, cuántos terminales hay y a qué velocidad se comunican, etc. Una vez configurados, estos parámetros establecen un nivel de coordinación que es deseable que sea óptimo. A continuación examinamos el núcleo y vemos las funciones que son necesarias para garantizar la integridad del sistema.

EL NUCLEO DEL SISTEMA OPERATIVO

El núcleo del sistema operativo es esa cantidad mínima de código esencial para su funcionamiento y que es usada intensivamente por todos los programas situados en niveles superiores. Su existencia es una buena muestra de una solución de compromiso. Un sistema operativo complejo está compuesto por más código del que cabe en la memoria de un ordenador, incluso del más grande. La creación del núcleo es el método que emplean los programas del sistema para utilizar la memoria disponible y repartirla entre el sistema operativo y los programas del usuario. La aceptación de esta solución implica que se deben resolver otras dos cuestiones del diseño: qué funciones deben residir en el núcleo y cómo deben interrelacionarse. Los núcleos pequeños proporcionan una información limitada de los procesos que están ejecutando; los núcleos grandes deben proporcionar servicios más amplios, como es la supervisión entre *procesos* que realizan llamadas y los que son llamados. Un núcleo realmente amplio debe

manejar también la *sincronización* de procesos, *distribución* de recursos y *manejo del directorio* del sistema. Los sistemas operativos más grandes y potentes pueden necesitar núcleos con tamaños superiores a 50 Kbytes. Dado que una función importante de los sistemas operativos es la de controlar, su diseño y, por tanto, el del núcleo, tiende a ser jerárquico (véase figura 2) en el sentido de que las funciones sólo llaman a otras situadas por debajo de ellas en la jerarquía.

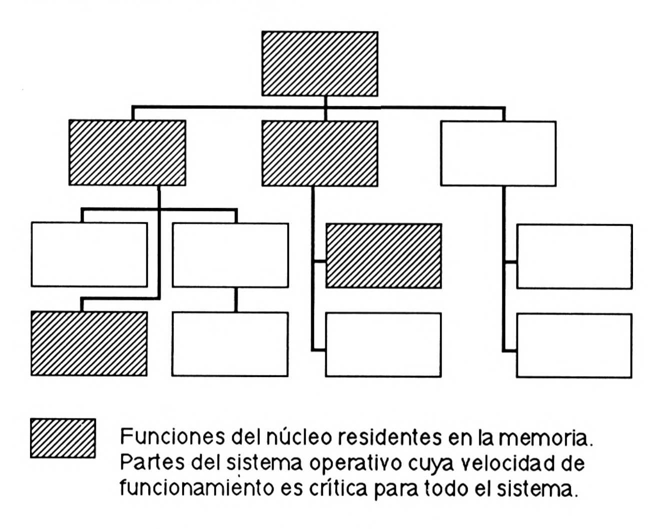


Figura 2. *Jerarquía de los módulos funcionales del núcleo.*

Esta política ayuda a identificar qué funciones conviene que estén en el núcleo. El resto de este capítulo está dedicado a describir las funciones esenciales del núcleo de un sistema mediano: manejo del almacenamiento, del procesador, de los dispositivos y de los procesos. No obstante, a estos as-

pectos del núcleo se les han dedicado capítulos separados en los que se discuten con más profundidad.

Manejo del almacenamiento

El almacenamiento primario o principal es un recurso esencial necesario para cualquier programa. El núcleo, desde luego, reside permanentemente en parte de este almacenamiento y el resto se divide en varias partes que se asignan a los programas del usuario. ¿Cómo se hace que estos programas funcionen sin molestarse unos a otros o, lo que es peor, sin molestar al núcleo? También debe considerarse una situación compleja en la que cambian constantemente los programas residentes en el almacenamiento. ¿Cómo se puede controlar a los programas del sistema encargados de cargar y conmutar entre diversos programas para que la seguridad no se vea afectada? Diferentes políticas conducen a diferentes mecanismos o soluciones. Por ejemplo, una política de manejo de almacenamiento que desee reducir los costes de implementación al mínimo utilizará un mecanismo de asignación de memoria *único contiguo* que admita sólo un programa de usuario en un momento dado. Otras políticas y mecanismos existentes:

Para decrementar la pérdida de tiempo y espacio asociada con la asignación única contigua se adopta el mecanismo de asignación *múltiple contiguo* que admite varios programas simultáneamente y los ejecuta concurrentemente. Esta política mantiene el procesador ocupado constantemente, pero conduce a la fragmentación del almacenamiento interno y externo.

Para reducir el impacto de la fragmentación de almacenamiento causado por los mecanismos contiguos se emplea un mecanismo de asignación *no*

contiguo. Este mantiene la ventaja de "el procesador ocupado" a la vez que elimina la pérdida de almacenamiento creada por la fragmentación de memoria.

Una vez que se han decidido las políticas de manejo de almacenamiento que se van a emplear, es necesario determinar las técnicas de control de espacio o de manejo de servicios. Esto se denomina genéricamente *asignación de almacenamiento dinámico;* se aplican mecanismos similares al manejo del almacenamiento externo para mejorar la utilización del espacio en disco. En el capítulo 3 se realiza un análisis más detallado del manejo del almacenamiento.

Manejo del procesador

Del mismo modo que todos los programas utilizan el almacenamiento como un recurso, también deben utilizar la unidad central de proceso (CPU) como si fuese otro recurso; por tanto, también hace falta optimizar el uso de ésta. El procesador se emplea para ejecutar los programas —programas que durante su ejecución se denominan procesos—. En base a esto se define un proceso como una unidad de código a la que se le asigna el procesador durante un tiempo. Los procesos los pueden originar los usuarios y el sistema operativo. Su control se ve complicado por los sistemas multiprocesador, en los que al proceso se le pueden asignar dos o más CPU. Las esferas de interés del manejo del procesador se limitan actualmente a tres funciones básicas (en el núcleo):

- *Control de procesos:* Asigna la CPU al proceso en espera de prioridad más alta y lo pone en marcha.
- *Preparación de trabajos:* Asigna los recursos necesarios en una tarea (una serie de procesos) antes de que ésta se ponga en la cola de espera

de ejecución. Normalmente estos recursos son archivos y dispositivos (y, naturalmente, almacenamiento).

- *Manejo de interrupciones:* Controla la comunicación entre los procesos que se están ejecutando, el subsistema de entrada/salida y el control de procesos. La interrupción impone una lógica externa a los procesos que se están ejecutando de modo que los recursos se pueden asignar de una forma eficiente y controlada. Las interrupciones se explican detalladamente en el capítulo 4.

Dado que el manejo del procesador, al igual que el manejo del almacenamiento, ofrece servicios de control a los procesos que se están ejecutando, es muy importante diseñar sus partes funcionales con el mayor cuidado posible para minimizar el tiempo que se dedica a estos procesos. Las funciones usadas frecuentemente son las que reciben una prioridad más elevada y deben tardar menos tiempo en ejecutarse que las funciones menos usadas. La aplicación de este principio al manejo del procesador se denomina *distribución multi-nivel.* El control de procesos se encuentra en el nivel más superior (frecuentemente usado), el mantenimiento de las colas de espera en un nivel intermedio y la recalculación de las prioridades en función de la actividad de entrada y salida se encuentra en el nivel inferior (usado con menos frecuencia).

Manejo de dispositivos

El número de tipos de dispositivos diferentes que se pueden conectar a un sistema para la entrada/salida de datos crece con el tiempo. El uso ordenado de dichos dispositivos es una tarea importante de los sistemas

operativos y todos disponen de un subsistema de entrada/salida para manejarlos. La sección dedicada al manejo de dispositivos examinará con más detalle el funcionamiento del subsistema de entrada/salida.

Los dispositivos se usan para transmitir información a y desde partes del sistema; estos dispositivos tienden a clasificarse como de *caracteres* o de *bloques*. Las actividades internas de agrupamiento en bloques y manejo de *buffers* se relacionan con el uso de dispositivos. Para poder realizar una descripción del tema del manejo de dispositivos se definen a continuación diversos tópicos, aunque se comentarán ampliamente en el capítulo 5.

Organización del hardware: Cuando se quiere conectar un dispositivo al sistema hace falta que disponga de unas señales de control que puedan gobernar su funcionamiento. Las señales transmitidas por el procesador se decodifican con una unidad de control (el controlador) que, a su vez, transmite las señales que el dispositivo es capaz de reconocer.

Proceso de peticiones: Cada proceso que se ejecuta genera peticiones de entrada/salida que necesitan ser atendidas por el sistema. El manejo de dispositivos interviene en el manejo de peticiones tales como la asignación de espacio de almacenamiento o el derecho a usar un periférico.

Acciones de entrada/salida: Debido a las importantes diferencias de tiempo de operación existentes entre el procesador y los dispositivos, hace falta el uso de áreas denominadas *buffers* situadas en la memoria principal y sirven como almacenamiento intermedio. En estas áreas se agrupan varios registros en bloques y hace falta desarrollar un sistema eficiente

que controle el manejo de cada archivo por medio de bloques y *buffers*.

Recuperación de errores: El punto más propenso a fallos de todo el sistema lo constituyen los dispositivos del subsistema de entrada y salida. Los mecanismos de *hardware* y *software* se emplean para detectar y, si es posible, corregir los errores que se puedan producir.

Manejo de procesos

Todo proceso que se crea en memoria debe ser tratado como un pseudo-recurso. Al contrario que los recursos reales son unidades efímeras que mueren una vez completada su misión. Mientras existen deben rendir un servicio y si no se los maneja correctamente pueden originar una gran degradación de los datos en calidad y/o seguridad. A continuación se nombran algunos de los conceptos relacionados con los procesos, pero la descripción en profundidad se realiza en el capítulo 6.

Concurrencia: Dos o más procesos que se ejecutan simultáneamente son concurrentes. Los procesos concurrentes pueden ser independientes o interactivos.

Exclusión mutua: Prohibe la ejecución concurrente de dos instrucciones que acceden a la misma área de memoria principal. Este área puede contener el valor de una variable compartida, en cuyo caso sí se permite el acceso.

Sincronización: Dos procesos que se excluyen mutuamente en el acceso a las variables compartidas siguen teniéndo que comunicarse entre sí para informarse mutuamente de sus estados. En este caso se emplean señales de sincronización para conseguir la coordinación los procesos *cooperativos*.

Comunicación: Si la información que se transmite entre los procesos cooperativos es algo más que señales de sincronización, por ejemplo el envío y la recepción de datos arbitrarios, se utiliza la comunicación entre procesos. Esto hace necesario la creación de un *buffer* de mensajes.

Interbloqueo: Normalmente se causa por dos o más procesos, cada uno de los cuales espera un recurso que posee uno de los otros. Debido a que están esperando, no llegan nunca al punto en que liberan su recurso y se le puede asignar a otro (situación mortal para el sistema).

En resumen, los procesos concurrentes pueden ser competidores o cooperadores; en este último caso se pueden dividir entre concurrentes o secuenciales. Una función importante del núcleo es la de vigilar y controlar las peticiones de recursos que realizan los diversos procesos, de modo que la seguridad y respuesta del sistema se conserven.

3

Manejo del almacenamiento

La arquitectura convencional de los ordenadores implica la existencia de un área de almacenamiento principal en la que se cargan los programas y sus datos anexos antes de ejecutarse. Incluso aunque en un momento dado sólo un 1 por 100 de este área esté en uso, el área de almacenamiento principal es imprescindible para que las instrucciones sean accesibles por el procesador. Cuando el recurso formado por la memoria principal se usa en la ejecución de un programa, no puede estar disponible simultáneamente para otro programa. Si hay otros programas que desean acceder a la memoria principal, se les mantiene normalmente en espera en un área de almacenamiento de reserva que en otro capítulo tratamos como si fuese un recurso separado. Una parte del almacenamiento principal se usa perma - nentemente para contener la porción residente del sistema operativo, el núcleo que hemos comentado en el capítulo anterior. Cada usuario debe asumir que el núcleo y el resto del *software* de soporte son correctos e inmunes a los funcionamientos incorrectos de los programas de los usuarios, ya que se ejecutan bajo su

control. Resulta fácil que un programa de un usuario intente salirse de los límites prescritos por el sistema operativo. En esta situación el núcleo debe funcionar correctamente para mantener un aislamiento completo de cada programa de usuario que se está ejecutando e, incluso más importante, evitar el mismo el ser adulterado por los programas de los usuarios. Estas funciones pueden llegar a ser muy complejas si se implementan ciertas políticas para optimizar la asignación de almacenamiento, ya que hacen que el diseño de este apartado del núcleo sea muy complejo. En esta sección estudiaremos, primero, las políticas más simples para realizar la función general de manejo del almacenamiento y, posteriormente, las más complejas. En cada caso también se considerará cómo el almacenamiento de reserva ayuda al mecanismo particular empleado para implementar dicha política.

ASIGNACION UNICA CONTIGUA

La asignación única contigua del almacenamiento es un mecanismo que implementa la política de conservar los costos de manejo del almacenamiento en un mínimo. Esto se realiza admitiendo un solo programa de usuario simultáneamente en memoria, lo que implica que los usuarios desfilan en serie por el sistema. Estas técnicas se usaron en los primeros sistemas operativos y son las que emplean los microordenadores actuales. Cuando se emplea este mecanismo el núcleo ocupa la parte inferior de la memoria (o la superior) y el sobrante se asigna al programa del usuario. Es decir, hay una sola cantidad de almacenamiento contigua disponible para asignar al programa que se va a cargar. Este mecanismo

presenta ciertas ineficiencias: El tiempo de procesador se desperdicia cuando el programa activo está esperando que se complete una operación, como es el escribir un bloque a disco. Este tiempo de procesador se pierde debido a que no hay otros programas esperando para usar el procesador. También se pierde todo el área de memoria que no use este programa. Como contraposición hay que considerar que la programación de esta función es relativamente simple: no hace falta elaborar un sistema de protección del almacenamiento (de otros programas), sólo hace falta un sencillo cargador de programas (como se explica en el siguiente apartado) y el manejo de la cola de trabajos en espera de su carga en memoria y ejecución es muy sencilla. Durante muchos años casi todos los sistemas de proceso de datos de tamaño medio y grande funcionaron perfectamente bien usando mecanismos de asignación única contigua para manejar el sistema.

CARGA Y RELOCALIZACION

La *carga* es el proceso de almacenar el programa y los datos en posiciones específicas de memoria antes de la ejecución del programa y se ejecutan con un cargador. Dado que nada puede suceder hasta que se cargan y ponen en marcha los programas, la carga es una función esencial necesaria para todos los programas y se incluye dentro del núcleo del sistema operativo. La terminación de un trabajo provoca una llamada al monitor de trabajos, que a su vez hace una llamada al cargador para que lea el siguiente trabajo que se encuentre en la lista de espera. Una vez que se ha realizado todo el proceso de carga, el cargador no es necesario y se realiza una llamada a la dirección de inicio del programa. El número de operaciones de

lectura que debe ejecutar el cargador se ve reducido si se agrupa el texto del programa en unidades grandes, cada una de las cuales ocupa posiciones de memoria consecutivas. Muchos programas de usuarios necesitan emplear *rutinas de librería,* por lo que éstas deben escribirse de modo que puedan ejecutarse en diversos puntos de la memoria. Esto se consigue con un mecanismo denominado *relocalización.* Un cargador relocalizable coge las librerías de rutinas y al realizar el proceso de carga establece una equivalencia entre posiciones relativas de almacenamiento y posiciones físicas de almacenamiento.

Los traductores producen segmentos de código relocalizable que, como su nombre indica, puede relocalizarse durante el proceso de carga. Todas las direcciones del código objeto son relativas al inicio del segmento —que puede ser cualquier posición arbitraria fijada por el monitor de trabajos— y, al cargar, todas estas direcciones relativas son reemplazadas por los números de posición física apropiados al segmento que se le ha asignado.

Si se segmenta el programa objeto es posible realizar un uso más sofisticado de la asignación contigua. Un segmento que no es necesario se puede guardar en el almacenamiento necesario hasta que haga falta, y cuando sea necesario se carga encima de uno o varios segmentos que hayan dejado de ser útiles. Esta variación se conoce como *carga dinámica,* dado que la decisión de cargar o no un segmento se realiza dinámicamente durante la ejecución del programa.

ASIGNACION MULTIPLE CONTIGUA

El costo de manejo de un sistema de asignación única contigua es mínimo, pero acarrea una pérdida de tiempo de procesador y de almacenamiento principal. Para emplear esta capacidad que se desperdiciaba se creó la *multiprogramación,* que permite que varios programas utilicen el almacenamiento principal simultáneamente y se ejecuten concurrentemente. Este mecanismo implementa la política de tener al procesador ocupado: mientras un programa espera que se complete una operación de entrada/salida, el procesador cambia a otro programa que esté esperando para ejecutarse. El espacio que se salva no es tan destacado, pero es generalmente mejor que la media con asignación única. Cada programa ocupa una región del almacenamiento y debe ser protegido de otros que ocupen sus propias regiones.

La región se construye con un conjunto continuo de posiciones de memoria y las diferentes políticas pueden requerir que las regiones sean fijas o que sean variables, permitiendo el crecimiento. Si la región se compone de *particiones fijas,* entonces se suele desperdiciar cierta cantidad de almacenamiento (como sucede con la asignación única contigua) y se produce una *fragmentación* de la memoria principal (véase figura 3). En este caso cada región puede estar asociada con su propia cola de trabajos y cargador; esto produce un equilibrado de las colas de trabajo si se conocen por anticipado los recursos que necesita cada programa.

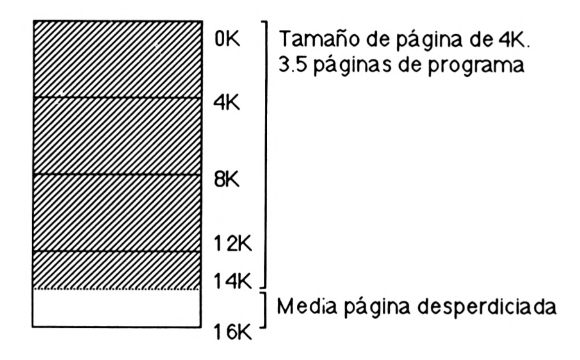


Figura 3. *Fragmentación de la memoria principal*

La fragmentación se puede reducir usando particiones móviles (o regiones rígidas variables) que ajusten el almacenamiento al tamaño exacto del programa. Esto requiere un manejador de espacio que manipule una tabla que contenga los valores de las regiones liberadas por trabajos que han terminado y asignadas a trabajos que están en marcha. Si los programas se pudiesen relocalizar dinámicamente hasta unirse con particiones de otros programas, entonces se podrían unir los fragmentos sin usar para crear espacio para otros programas. Si se usan *regiones móviles* y se introducen durante la ejecución del programa, entonces se emplearía la *relocalización dinámica* y se produciría una optimización global de toda la memoria asignada a programas. Si se emplea almacenamiento secundario para ayudar al movimiento de programas entre regiones, entonces se está aplicando el concepto de *swapping*, lo que implica que el manejador de espacio tiene que llevar una lista de los trabajos que se encuentran en la memoria principal y en disco.

ASIGNACION NO CONTIGUA

Si no existe la condición de que el programa debe residir en un conjunto consecutivo de posiciones físicas de almacenamiento, entonces se pueden utilizar los fragmentos que de otro modo se hubieran desperdiciado. En este caso el usuario puede ignorar en qué parte exacta de la memoria se encuentra el programa, ya que existen mecanismos que transforman las direcciones físicas en otras que sean utilizables por el programa.

La adición al sistema de técnicas de *paginación* o segmentación proporciona dos ventajas: los programas pueden funcionar como si existiera bastante más espacio del que realmente existe y se puede efectuar una asignación de posiciones virtuales de memoria a posiciones reales que no tienen por qué ser contiguas. La paginación depende de la organización de la memoria en un grupo de *bloques fijos* o páginas.

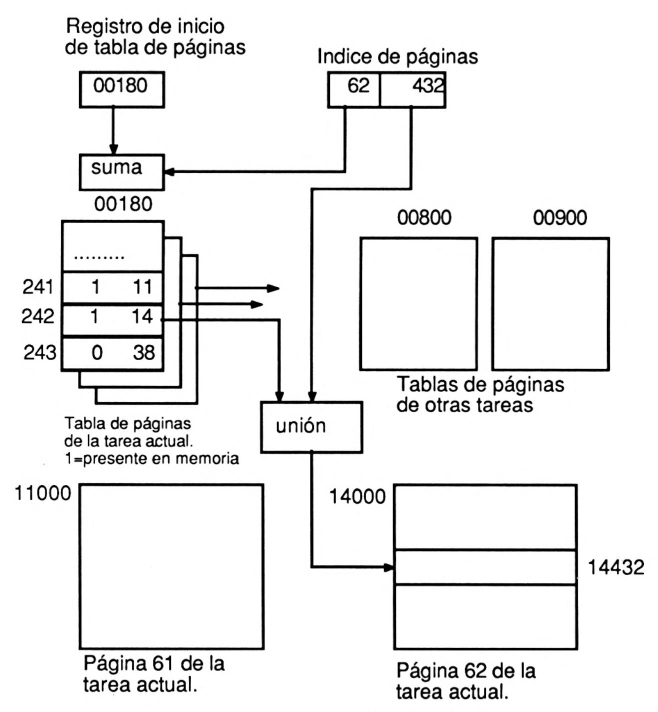


Figura 4. *Paginación*

La fragmentación puede ocurrir dentro de las páginas (fragmentación interna), pero si el tamaño de las páginas es de 2K a 4K esta fragmentación es relativamente pequeña (véase figura 4). Algunos manejadores de espacio trabajan agrupando las páginas para minimizar el tiempo empleado en el manejo de éstas.

La división del programa en páginas se realiza por medio de un traductor que no realiza ningún agrupa -

miento lógico de instrucciones o datos. El almacenamiento secundario guarda cualquier número de páginas extra que se paginan en memoria cuando haga falta. Las instrucciones son contiguas dentro de una página y el mecanismo de paginación transforma una memoria real no contigua en memoria virtual contigua. La *segmentación* hace que el programa se divida en segmentos lógicos de longitud variable, cada uno de los cuales recibe un nombre de segmento. El almacenamiento es contiguo dentro de un segmento, pero los segmentos no tienen por qué usar posiciones contiguas de memoria. La segmentación también facilita la compartición de recursos: en un entorno de desarrollo de programas se usan frecuentemente compiladores reentrantes. El segmento del compilador que se está usando se introduce en las tablas de segmentos de los usuarios que los están usando. Sólo hace falta actualizar la dirección de la tabla si el segmento se borra y se vuelve a cargar. Cuando la segmentación se combina con los mecanismos de paginación, se obtienen ganancias significativas. La segmentación paginada alivia el problema de la fragmentación residual y simplifica el uso de almacenamiento secundario, pero para evitar retardos intolerables se necesita la ayuda del *hardware*, normalmente en la forma de una unidad de manejo de memoria (MMU).

4

Manejo del procesador

Ya se ha indicado previamente que al procesador se le trata como si fuese otro recurso, dado que se debe garantizar que el programa recibe su servicio para poder avanzar. Por tanto, si se quiere que el sistema funcione correctamente y con todos sus componentes equilibrados, hay que hacer que el manejo del procesador reciba la misma atención.

El procesador ejecuta una instrucción de un programa cada vez. Mientras el programa se está ejecutando existe un proceso. En un contexto de multiprogramación pueden existir varios programas ejecutándose concurrentemente en cuyo caso existirá un número igual de procesos en progreso, incluso si cada proceso es una versión reentrante del mismo programa, y una tarea de un usuario puede llamar a dos o más procesos antes de terminar. Se puede desarrollar más esta definición para decir que un *proceso* es una unidad a la que se le asigna el procesador.

Del mismo modo que tratábamos el tema de la memoria virtual y real, que permitía que diferentes procesos ocupasen las mismas posiciones de memoria

simultáneamente, podemos imaginar un solo procesador ejecutando operaciones de un modo independiente a otros procesos que tengan demandas similares del mismo procesador. Dicho de otro modo, diferentes procesos parecen ejecutarse concurrentemente usando sus procesadores virtuales.

Hacer que el sistema funcione de esta forma implica una penalización en tiempo, dado que un solo procesador asignado a un proceso terminará ese proceso antes que el mismo procesador asignado a, por ejemplo, dos o más procesos idénticos, dado que el número de instrucciones a procesar aumentará en la misma proporción que el número de procesos asignados. Esta sección planteará un modelo de *asignación de procesador* antes de considerar los aspectos de control de procesos, manejo de tareas e interrupciones.

MODELO DE PROCESADOR

Vamos a desarrollar un modelo de un sistema de un solo procesador capaz de multiprogramación. Para conseguir esto postulamos que cada proceso puede estar en uno de tres estados posibles y sólo un procesador puede estar ejecutando instrucciones. El proceso que se esté ejecutando decimos que está en *estado de ejecución*. Cualquier proceso que pueda ejecutarse tan pronto como el procesador quede disponible decimos que está en el *estado de preparado* y tiene todos sus recursos asignados con excepción del procesador. Los procesos que han parado su ejecución esperando a que se complete otra actividad se encuentran en un *estado de bloqueo*. Sólo puede haber un proceso en estado de ejecución; todos los demás procesos activos están o en bloqueo o en preparado. Un proceso puede dejar el estado de eje -

cución debido a su terminación o a que ha solicitado un recurso, por ejemplo un proceso de entrada/salida, y se ha bloqueado esperando la terminación de una actividad, o a que el sistema operativo lo ha parado o borrado debido a que el proceso ha excedido algunos límites (por ejemplo, un tiempo máximo). Un proceso bloqueado o dormido es despertado por el sistema operativo cuando termina la espera y se coloca en el estado de preparado; al cabo de un tiempo el sistema operativo lo pone en estado de ejecución transfi - riéndole el control (véase figura 5).

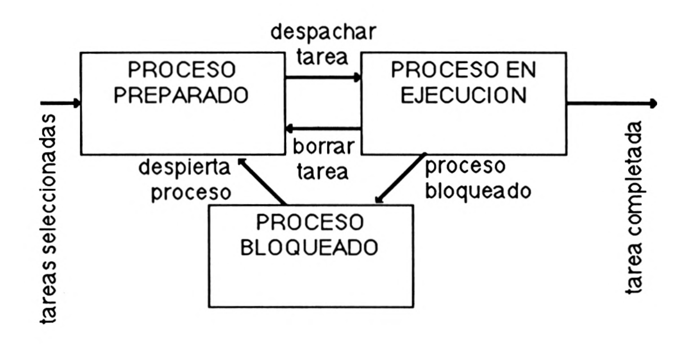


Figura 5. *Estados de tareas y procesos*

El manejo de tareas se realiza creando un proceso y colocándolo en estado de preparado después de haberle asignado todos los recursos necesarios con excepción del procesador. A continuación la rutina encargada del control de procesos le asigna el procesador y el proceso conserva éste hasta que se bloquea. Las rutinas de manejo de tareas y de control de procesos están examinando continuamente colas de tareas y procesos preparados, ya que éstos cambian constantemente. Una tarea sólo se introduce en la cola de tareas una vez, mientras que puede ser necesario poner un proceso en estado de ejecución varias veces

antes de que se complete. Puede considerarse que este modelo de procesador es capaz de ejecutar varios procesos simultáneamente con un solo proceso real en el control en un momento dado y el resto de los procesos considerados *virtuales*.

CONTROL DE PROCESOS

Todos los procesos virtuales que se encuentren en el estado de preparado pueden ponerse en marcha de acuerdo con su orden de prioridad. La función encargada del control seleccionará el proceso preparado con la más alta prioridad y le asignará el recurso del procesador. Los factores que determinan la prioridad pueden variar de un sistema a otro; sin embargo, casi todos los sistemas usan un *registro de descripción de procesos* que se emplea para determinar la prioridad. Este registro puede contener información relativa al tiempo transcurrido desde el último servicio, tiempo empleado en el último servicio, momento de su creación, etc. A continuación vamos a examinar algunas políticas empleadas y los mecanismos empleados para implementarlas.

Una política consiste en asignarle a todos los procesos el mismo tiempo medio de espera de servicio. Un sencillo mecanismo compuesto por una sola cola puede implementar un servicio del tipo primero-en-llegar—primero-servido. El inconveniente de este sistema es que el tiempo medio de espera se incrementa de un modo inaceptable cuando se crean muchos procesos; también se incrementa en exceso si la duración de los procesos varía considerablemente. Una variación del sistema anterior es la denominada "siguiente trabajo más corto", que emplea la duración del proceso en lugar de su creación para determinar la prioridad. Otra variación sencilla que

evitará la duración excesiva de determinados procesos es la de "rebanadas de tiempo"; en ésta se le asigna un determinado tiempo a cada proceso, al final del cual se para el proceso y se pone al final de la cola, poniéndose en marcha el siguiente proceso que se encuentre en estado de preparado. Los nuevos procesos se colocan al final de la cola. Este mecanismo es muy popular en sistemas interactivos, siendo el más usado.

Diseños más complejos permiten la existencia de múltiples colas para realizar el control del flujo de los procesos. Cada cola puede tener su propio mecanismo control de tareas diseñado especialmente para reflejar la diferente política de servicios de esa cola. En una asociación dinámica de colas los procesos no están obligados a permanecer en la misma cola, sino que pueden cambiar de una a otra en función de la realimentación recibida de los procesos en ejecución. Otro sistema que se emplea es el control de terminación de procesos. En este caso se intenta garantizar la terminación de un proceso. Una aproximación sencilla usa una cola de procesos preparados ordenados ascendentemente por prioridad de terminación; un mecanismo más complejo atiende a los procesos en tiempo real, interactivo y de lotes de un modo distinto para asegurar que cada categoría recibe un porcentaje definido de tiempo de procesador.

PREPARACION DE TAREAS E INTERRUPCIONES

La función de la rutina encargada del manejo de tareas es asignar recursos a los procesos que entren en *estado de ejecución* y recuperar los recursos de un proceso que ha terminado o está bloqueado. Entre dichos recursos se encuentran archivos, dispositivos y

almacenamiento ejecutable. La rutina de manejo de tareas lleva una lista de todos los recursos liberados cuando termina un proceso, de modo que es capaz de determinar cuándo puede ser seleccionada otra tarea para uso de un procesador virtual (véase figura 6). Del mismo modo que con el control de procesos (apartado anterior), en este caso también se puede emplear un servicio del tipo primero-en-llegar—primero-servido o siguiente trabajo más corto para determinar cuándo se deben asignar archivos, dis - positivos o almacenamiento a una tarea que espera un procesador virtual.

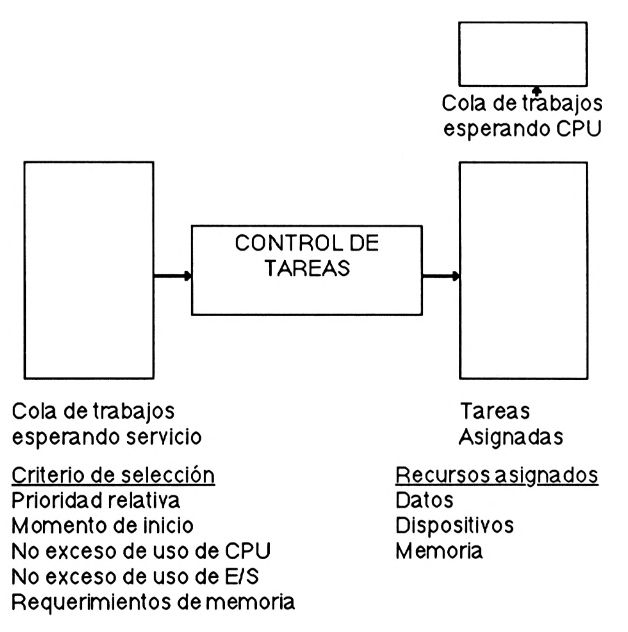


Figura 6

Si no se utiliza el sistema de primero-en-llegar—primero-servido, se puede emplear uno que aumente

la prioridad de una tarea cada vez que es considerada pero no puesta en marcha; de este modo se asegura su ejecución en algún momento. Este control puede ser extremadamente difícil de realizar debido al creciente número de relaciones entre peticiones de recursos y disponibilidades resultantes de las consideraciones de diseño que afectan a ambos, el manejo del procesador y a todo el sistema operativo.

Todos los mecanismos de control de procesos y manejo de tareas se basan en la realimentación de los procesos activos; esta realimentación toma la forma de una interrupción. Por ejemplo, la expiración de una rebanada de tiempo o la terminación de una actividad de entrada y salida pueden causar una interrupción que haga que se replantee la situación. Las interrupciones son externas si son causadas por procesos asíncronos que ocurren externamente al procesador interrumpido, como las generadas por el subsistema de entrada/salida. Son internas si su causa se asocia con el proceso que se está ejecutando, como un fallo de una página, desbordamiento aritmético o petición de memoria fuera de rango. Parte del proceso de interrupciones es manejado por *hardware* y parte por una función del sistema operativo denominada *rutina de interrupciones* que es parte del núcleo del sistema. Pero, ¿qué sucede si se produce una interrupción mientras se está atendiendo una anterior? Las interrupciones múltiples se manejan con mecanismos de prioridad. La prioridad refleja el tipo de interrupción; por ejemplo, una llamada al supervisor puede recibir una prioridad superior que una interrupción de entrada/salida. La rutina de interrupciones necesita tener colas de acceso en las que se almacenen los detalles de las peticiones de interrupción pendientes de procesar.

5

Manejo de dispositivos

Los modernos sistemas de ordenadores son capaces de manejar una gran cantidad y diversidad de dispositivos que manejan la entrada/salida y el almacenamiento secundario. La utilización correcta de estos dispositivos, sean usados frecuentemente o no, es una de las principales tareas del sistema operativo. El *manejo de dispositivos* es responsable de los aspectos de organización de ficheros en conexión con los *buffers* y la organización de los datos en bloques, así como la respuesta del subsistema de entrada y salida que maneja la comunicación y acceso hacia y desde los dispositivos.

Esta sección estudia cómo se organiza el *hardware* con respecto a la comunicación, controladores y dispositivos; cómo se realizan los accesos a estos dispositivos para obtener una eficacia máxima; cómo se agiliza la transmisión de datos entre el almacenamiento principal y otros dispositivos mediante el empleo de *buffers,* y qué medidas se pueden tomar para manejar las situaciones de error que se produzcan durante el funcionamiento.

Desde que los sistemas operativos como el UNIX están disponibles en el mercado, los diseñadores de sistemas han tendido a ver los dispositivos como programas o procesos que simulan un dispositivo dado. Si se consigue esto, entonces las técnicas de sincronización que se han aplicado a los procesos pueden aplicarse también a los dispositivos que se han implementado como procesos. Los sistemas operativos del futuro se diseñarán, casi con toda certeza, usando *dispositivos virtuales* que se definen como procesos que simulan un dispositivo existente pero no disponible en ese momento. Otro beneficio importante de los dispositivos virtuales es la "operación simultánea de diversos periféricos en linea", también conocida como *spooling*, y que es el aparente funcionamiento concurrente y compartido de dispositivos secuenciales de entrada/salida, en especial impresoras.

ORGANIZACION DEL HARDWARE

La suma total de todas las posibles acciones físicas que todos los dispositivos conectados al procesador son capaces de realizar exceden el número de códigos internos existentes en el sistema. Por tanto, las señales de control que activan un dispositivo particular provienen de una *unidad de control* que interpreta un rango válido de órdenes que le llegan del procesador. También puede existir una unidad de control encargada de dos o más dispositivos y que incluso pueden ir empaquetadas con los dispositivos. Normalmente los dispositivos son bastante lentos y asíncronos con respecto al procesador. La unidad de control recibe sus instrucciones del procesador por

medio de un *canal* que trabaja a la velocidad del procesador. Las instrucciones especifican las operaciones a realizar por los dispositivos, incluyendo el control de los datos transferidos entre la unidad de control y la memoria principal. Un gran número de dispositivos lentos, como el teletipo, pueden manejarse multiplexando cada uno como si fuese un subcanal (canal multiplexor). Por tanto, se puede ver el subsistema de entrada/salida como varios dispositivos conectados a una unidad de control; varias unidades de control conectadas a cada canal, y varios canales conectados a la memoria principal del procesador. Existen dos técnicas para manejar las comunicaciones entre el procesador, las unidades de control y los dispositivos: proceso de interrupciones y *polling*. Por un lado el canal puede generar una *interrupción* al procesador que responderá dependiendo de la prioridad de la interrupción; alternativamente el procesador puede interrrogar *(poll)* a todos los canales periódicamente para ver si alguno necesita su atención. El *polling* puede ser una pérdida de tiempo si no hay ningún canal esperando ser atendido, pero, dado que el *polling* sucede sólo cuando el procesador queda libre, no es una gran desventaja. La distinción entre *polling* e interrupción es quizá más fina, ya que para que una interrupción funcione en un ordenador es necesario que el procesador examine *(poll)* determinados bits por *hardware* entre cada dos instrucciones para ver si hay petición de interrupción.

PROCESO DE PETICIONES

Casi toda la actividad del manejo de dispositivos está relacionada con el control de peticiones para transferir datos entre la memoria y un dispositivo o entre un dispositivo y otro por medio de la memoria.

Al manejar esta situación el sistema operativo debe controlar, al menos, tres aspectos importantes: la asignación de espacio en el dispositivo indicado, el derecho a usar la vía de comunicaciones hacia dicho dispositivo (seguridad de acceso) y el control del funcionamiento del dispositivo.

Asignación de espacio

Los dispositivos tales como los discos se pueden compartir entre varios usuarios. Es importante que no se les permita a los usuarios acceder directamente a un dispositivo compartido para evitar un uso ineficiente de la unidad y la interferencia con datos de otros usuarios. En consecuencia, un componente del sistema operativo, denominado a veces supervisor de entrada/salida, controla el acceso a todos los dispositivos compartidos (en realidad a todos los dispositivos). El supervisor debe asignar espacio en el dispositivo antes de que el usuario pueda acceder.

Asignación de acceso

Existen varios sistemas para realizar una petición de acceso a disco. Las técnicas usadas intentan optimizar la velocidad de transmisión hacia y desde el disco y normalmente se encuentran incorporadas en la rutina de manejo de disco. Una política emplea el sistema del menor tiempo de espera primero, que guarda una lista de todas las peticiones para minimizar el movimiento del brazo del disco (la tarea más lenta de todo el acceso). Esto puede provocar que las peticiones de datos situados en cilindros alejados se vean considerablemente retrasadas. Otra política prohibe la dirección inversa del brazo mientras haya una petición insatisfecha en la dirección actual de movimiento.

Esto provoca movimientos alternos de la cabeza que puede posponer peticiones antiguas en favor de otras nuevas si éstas se encuentran en la dirección actual de la cabeza.

Funcionamiento del dispositivo

Otra obligación de la rutina de manejo del disco es controlar la operación del dispositivo iniciando su ejecución después de comprobar que dicho dispositivo está asignado al usuario. El sistema operativo UNIX usa un subsistema de entrada y salida que trata cada dispositivo como si estuviese formado por bloques de 512 bytes direccionables aleatoriamente (en los dispositivos orientados a bloques) y otro subsistema de entrada y salida para manejar datos como flujo de caracteres (dispositivos orientados a caracteres).

ACCIONES DE ENTRADA/SALIDA

Al discutir la transferencia de datos entre la memoria principal y los dispositivos externos hemos de tomar nota de:

a) Las diferencias de velocidades de operación entre la memoria principal y el dispositivo.
b) Los tiempos relativos de acceso y transferencia de datos.
c) El hecho de que la memoria principal y los dispositivos actúan asíncronamente.
d) El espacio *(gap)* existente entre registros en las cintas magnéticas.

Estos factores indican que los datos deben haberse leído bien antes de poder ser manejados por el programa y deben escribirse bien después de ser creados o actualizados por el programa. Si el sistema maneja bloques grandes de información entonces debe hacer menos accesos. Los *buffers* de almacenamiento de datos se usan para contener bloques de datos. La transmisión de datos entre los *buffers* y los dispositivos es posible de forma concurrente con el procesado de otros datos. Para los propósitos de manejo de dispositivos, el archivo de datos se organiza como registros que se agrupan para la transmisión. Las operaciones de lectura transfieren bloques desde el archivo al *buffer* de entrada situado en la memoria principal. Las siguientes operaciones de lectura se refieren a los registros que se encuentran en el *buffer* de entrada, hasta que se procesa el último. Del mismo modo las operaciones de escritura escriben el área del *buffer* de salida del archivo en el dispositivo. Se deben buscar métodos para establecer e implementar los *buffers* y los bloques en cada fichero (véase siguiente apartado). Las peticiones intermedias de lectura y escritura de registros se satisfacen empleando los *buffers* de entrada y salida (véase figura 7).

MANEJO DE DISPOSITIVOS DE BLOQUES

Los sistemas convencionales de empleo de *buffer* utilizan un *buffer* de entrada y otro de salida que contienen uno o más registros, dependiendo del empaquetamiento que se emplee. También hace falta un *área de trabajo* para contener el registro que se esté procesando. Una vez procesado, el registro que se encuentra en este área de trabajo se copia en el *buffer*

de salida. Estos dos *buffers* tienen un puntero que indica cuándo tiene que leerse un dato nuevo o cuándo se debe escribir en disco antes de que pueda continuar el proceso. El sistema de *buffer* doble emplea dos *buffers* de entrada y dos de salida; de este modo, mientras un *buffer* de entrada se procesa, el otro puede recargarse con datos del disco, y así sucesivamente, de modo que las funciones que eje - cutan se alternan de unos a otros.

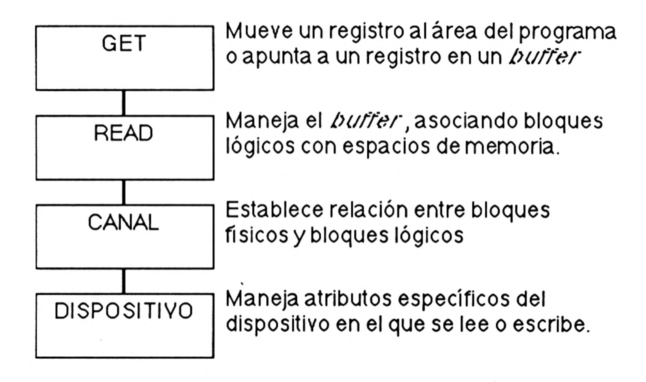


Figura 7. *Cuatro niveles de actividad de entrada/salida*

El sistema de *buffer indirecto* (que usa un sistema más complejo de punteros) utiliza un área dada de almacenamiento como *buffer* de entrada; a continuación, como área de trabajo y, finalmente, como *buffer* de salida antes de volver a empezar el ciclo. Las transferencias hacia y desde los *buffers* se realizan por medio de bloques de datos, y por esta razón los registros lógicos de un fichero se agrupan para formar un bloque (registro físico). El factor de bloqueo es el número de registros lógicos (fijados por el programa) que constituyen un registro físico. Cuando funcionando con este tipo de bloques un programa realiza una operación GET para transferir el

siguiente registro lógico del *buffer* de entrada al área de trabajo, puede suceder que el *buffer* de entrada esté exhausto; entonces la rutina de manejo de bloques realizará una función READ para rellenar el bloque de entrada con la transferencia de un bloque desde el disco. De un modo similar, cuando el programa ejecuta una instrucción PUT para transferir el registro lógico del área de trabajo al *buffer* de salida puede suceder que éste se encuentre lleno, en cuyo caso la rutina de bloques ejecutará una instrucción WRITE para grabar el *buffer* de salida en su espacio correspondiente en disco.

El *factor de bloqueo,* en los archivos organizados secuencialmente, determina la frecuencia con la que se ejecutan las instrucciones READ y WRITE. La distribución en bloques agiliza el procesamiento de registros lógicos, pero como contrapartida se tiene el espacio que ocupan los *buffers*. El supervisor de entrada/salida también se encarga de preparar los ficheros para su uso y de cerrarlos después de que hayan sido usados.

La instrucción OPEN prepara el terreno para: *a)* leer y comprobar la etiqueta del fichero (entrada) o escribir dicha etiqueta (salida); *b)* crear los *buffers,* y *c)* leer las rutinas necesarias para manejo de dispositivos y/o archivos.

La instrucción CLOSE es más sencilla y se utiliza para: *a)* escribir el último *buffer*, esté total o parcialmente lleno; *b)* liberar el área usada por los *buffers,* y *c)* escribir etiquetas de fin de archivo (en operaciones de salida).

RECUPERACION DE ERRORES DEBIDOS AL MAL FUNCIONAMIENTO DEL DISPOSITIVO

Dado que los dispositivos del sistema son de naturaleza mecánica, el trato físico así como la contaminación y los fallos humanos provocarán en algún momento un error de entrada/salida. Los componentes del *hardware* llevan incorporados muchos mecanismos de comprobación de error que evitan que dicho dispositivo sea usado hasta que se corrige el error. Las técnicas de grabación permiten la detección y corrección de los datos grabados (por ejemplo, la redundancia horizontal y vertical y la comprobación de paridad). Un sistema que resulta eficaz cuando se escriben archivos en discos es el de leer después de escribir para comprobar si los datos coinciden. Si se produce un error que no se puede corregir inmediatamente, entonces se usa una política de repetición. Si la operación se repite sin error, entonces la condición es *soft* y el proceso continúa. Este proceso puede ser totalmente transparente para el usuario que sólo se enterará si el proceso de repetición automática falla. Un fallo serio del *hardware* a la mitad de un proceso puede hacer necesario un procedimiento de recuperación que haga que el trabajo se vuelva a reinicializar después de corregir el fallo (por ejemplo, se estropea la fuente de alimentación o la cabeza de un disco toca la superficie).

Con el desarrollo de dispositivos que incorporan controladores más inteligentes por medio del uso de microprocesadores, muchos aspectos funcionales relativos a la detección de errores se están eliminando del sistema operativo. Hoy en día los errores causados

por equivocaciones humanas, como el instalar un volumen equivocado en un dispositivo, son mane - jados por el subsistema del dispositivo, que responde con los mensajes de error apropiados en la consola del sistema.

6

Manejo de procesos

Aunque hemos denominado a esta sección "manejo de procesos", no estamos tratando un recurso en el mismo sentido que el almacenamiento, los dispositivos o el procesador. En cierto sentido los procesos son consumidores de recursos en lugar de recursos por sí mismos; sin embargo, en otro sentido, una vez que han entrado en el estado transitorio de ser procesos pueden considerarse candidatos a ser *manejados*. Los procesos pueden rivalizar por los recursos existentes en un entorno multiprogramación hasta un nivel tal que llegue a causar efectos indeseables; este daño potencial se examina en los requerimientos de procesos concurrentes que interaccionan. La concurrencia, que se refiere a la ejecución solapada de procesos secuenciales, es la causa de un gran número de situaciones clásicas que se resuelven aplicando técnicas de manejo de procesos.

Un *proceso secuencial* se define como ejecución de un programa en un procesador secuencial. Si se puede implementar un mecanismo para que dos o más procesos tengan parte de sus procesos solapados en tiempo, entonces estos procesos se llaman *concurrentes*. Si cada proceso concurrente funciona con sus propios datos, y éstos no son manejados por otro

proceso concurrente, entonces los procesos son *di-juntos*. Si los procesos concurrentes comparten datos comunes entonces se dice que los procesos *inter - accionan*. Los procesos concurrentes pueden interac - cionar competitivamente (procesos compartidos) o cooperativamente (procesos comunicativos). Esta sec - ción examina los métodos por los que los procesos pueden:

a) Excluirse mutuamente unos a otros de acceder simultáneamente a los datos.
b) Sincronizar sus ejecuciones para conseguir dicha exclusión mutua.
c) Usar *buffers* de mensajes entre-procesos para conseguir la comunicación entre procesos simultáneos o sistemas multiprocesador.

EXCLUSION MUTUA, SINCRONIZACION Y COMUNICACION

Resulta imposible manejar varios procesos si se permite el acceso indisciplinado a los datos com - partidos. Para prevenir la ejecución incorrecta de procesos competitivos hace falta una *exclusión mutua* que garantiza que sólo un proceso accede simul - táneamente a los datos compartidos. Si los procesos cooperan, además de excluirse mutuamente de los datos compartidos, también deben hacerse *señales*. La menor unidad de transmisión será una señal de temporización. El primer proceso produce una *espera de evento* mientras aguarda a que se produzca un suceso; el segundo proceso produce una *señal de evento* para indicar la ocurrencia de dicho evento

sus procesos por medio de dichas señales. En su forma más sencilla se utiliza un solo bit como variable indicadora y el sistema sólo permite que haya un proceso esperando a que se produzca un evento dado. Este concepto puede aplicarse al control del acceso compartido a archivos.

Un sistema puede ofrecer las opciones: n lectores del archivo pero ningún escritor; un escritor pero ningún lector; un escritor y n lectores, y n lectores y n escritores. Se puede hacer que los procesos cooperen más estrechamente si se proporciona un mecanismo de comunicación consistente en el envío y recepción de datos llamados *mensajes*. En un sistema de un solo procesador los procesos cooperativos no se pueden ejecutar simultáneamente; en un sistema multiprocesador pueden ejecutarse, aunque no es necesario. Por esta razón existe un *buffer* de mensajes para contener éstos después que se han enviado pero antes de recibirse. Los *buffers* de mensajes pueden estar dedicados a determinados procesos o compartidos por todos; en ambos casos se produce la desventaja de que se crea otro recurso a manejar (véase figura 8). Algunos sistemas implementan la característica de los *buffers* de mensajes como *tuberías* que direccionan la salida de datos de un proceso a la entrada de otro; se puede emplear toda la cantidad que se quiera de tuberías para conectar procesos que se comunican en serie.

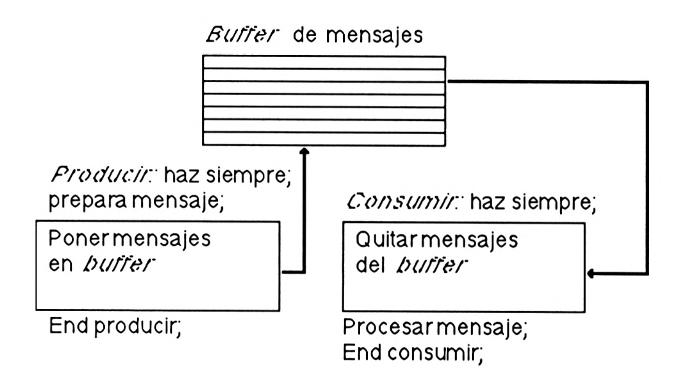


Figura 8. *Los procesos que cooperan producen y consumen mensajes que sincronizan*

INTERBLOQUEO

Un producto indeseable de la concurrencia es el *interbloqueo* —una situación en la que dos o más procesos se *bloquean* para siempre—. Esto puede ser causado porque cada uno de los dos procesos espera un recurso poseído por el otro. No se origina por un error de programación, sino por procesos que compiten por los recursos del sistema operativo. Para tratar el tema del interbloqueo se pueden considerar tres perspectivas distintas —prevención, evitación y detección—. La política de *prevención* asegura que las condiciones que pueden producir un interbloqueo no se producen nunca, pero esto puede llegar a ser muy difícil. *Evitación* no asegura que no se pueda producir un interbloqueo, pero intenta garantizar que no ocurrirá en las condiciones particulares del conjunto actual de procesos. Este mecanismo requiere un considerable conocimiento del proceso de petición de recursos por los procesos y puede resultar muy

complejo de implementar. Los implementadores que consideren que los mecanismos de prevención o evitación son demasiado caros en tiempo o espacio adoptarán la política de *detección*. En este caso se considera que la posibilidad de interbloqueo es lo suficientemente rara como para justificar el costo de un mecanismo que la detecte y recupere el sistema después de detectarla. Una vez que se ha realizado la detección (algunos sistemas obligan a que sea el operador el que lo haga) se debe recuperar el sistema. Normalmente esto requiere la reinicialización de uno o más procesos, si es necesario, desde el principio. Se puede emplear una técnica de chequeo de las usadas en recuperación de errores para recuperar el sistema de un interbloqueo. Una reinicialización segura terminaría todos los procesos y volvería a arrancar el sistema operativo, pero esta medida puede ser demasiado drástica en determinados ambientes. A veces se pueden detectar y romper los interbloqueos a través de un procedimiento de recuperación de modo que resulten invisibles para los usuarios. El costo de la política de detección depende de la frecuencia con que es necesaria, siendo el coste principal la degradación que se produce en tiempo.

7

Manejo de archivos

Los ordenadores, al igual que muchos otros desarrollos tecnológicos, se desarrollaron rápidamente cuando apareció un incentivo comercial para ellos. En los ordenadores el incentivo comercial fue su habilidad de procesar grandes volúmenes de datos almacenados que podían suponer un conocimiento estratégico y muy detallado del funcionamiento de la empresa. Después del descubrimiento del hecho de que los ordenadores de propósito general producidos para el comercio podían usarse también en las universidades y en los centros de investigación, se aceleró el desarrollo de diseños de segunda, tercera y cuarta generación. Hoy en día está claro que una de las principales facilidades de los ordenadores es su capacidad de almacenar grupos de archivos de datos. Si este recurso, muchas veces de muy gran tamaño, no se maneja tan eficientemente, al menos, como los otros recursos, la respuesta del sistema se degradará. El manejo de archivos de los sistemas operativos intenta hacer este aspecto, al menos tan eficiente como otros aspectos del manejo de recursos.

Existen dos razones principales por las que el manejo de archivos se incorpora al sistema operativo:

- Relevar al usuario de la tarea de almacenar los archivos fuera del ordenador haciendo que la tarea de recuperación sea más sencilla.
- Permitir a dos usuarios compartir el mismo fichero. El sistema operativo es una excelente posición para arbitrar a varios usuarios que deseen acceder al mismo fichero a la vez.

Para conseguir esto se diseñan los sistemas operativos de modo que implementen políticas que realicen: *a)* protección contra pérdida o corrupción; *b)* asignación eficiente del almacenamiento secundario, y *c)* un interfaz sencillo con el usuario que permita nombres de archivos simbólicos, atributos y compartición de archivos.

Esta sección describe brevemente cómo se manejan las librerías o archivos, cómo se organizan individualmente estos archivos (organización interna) y la jerarquía de funciones del sistema operativo que es necesario conseguir para implementar un manejo efectivo de los archivos y dispositivos.

FUNCION Y ORGANIZACION

El sistema de manejo de archivos ofrece un conjunto de funciones que puedan ser llamadas por el usuario (por ejemplo, crear un archivo), o realizarse de forma automática (borrar un archivo antiguo automáticamente al crear una nueva versión). Durante la operación normal del sistema, los usuarios, por medio de sus programas, necesitan realizar manipulaciones de alto nivel en los archivos. Necesitarán escribir (WRITE), leer (READ) y en otras situaciones podrán necesitar renombrar (RENAME), borrar (DELETE) o copiar (COPY) el archivo. Estas actividades, junto con algunas otras,

representan la lista de funciones del sistema de archivos que pueden ser llamadas por el usuario. El usuario se basa en el sistema de manejo de archivos para controlar el acceso a los archivos e implementar las otras políticas que se vayan a emplear (seguridad y eficiencia). En los grandes ordenadores el sistema de archivos utiliza una política de obtención de copias de seguridad y recuperación de datos automática, aunque los usuarios pueden hacerlo también cuando lo deseen. En la mayoría de los sistemas el mantenimiento automático y la orga- nización de la estructura del sistema de archivos también son indivisibles al usuario.

El sistema de archivo se organiza de modo que el acceso se realiza en dos etapas. En primer lugar se busca su nombre en el directorio y entonces (si existe) se busca el registro físico de datos que contiene la información que se quiere procesar. Si existe un nú - mero muy elevado de archivos en el disco, no resulta eficiente emplear un directorio *lineal* y se emplean directorios con estructura de *árbol*. Esta última ha demostrado ser muy eficaz y existen diversas implementaciones que dan prueba de su éxito (el UNIX es una de ellas). En el caso de estos directorios, una entrada en un directorio puede ser el nombre de un archivo u otro directorio (subdirectorio), razón por la que recibe el nombre de estructura de árbol. Por tanto, cada vez que se añade un nuevo subdirectorio se amplía el nombre de los ficheros de ese sub - directorio al añadirle otro prefijo. Por ejemplo, /u/Juan/física/informe indica, si el informe es el archivo, que todos los demás son directorios, ocupando el nivel superior /u. Los sistemas de manejo de archivo que están organizados en directorios con estructura de árbol ofrecen mayor facilidad en la manipulación de los directorios, existiendo funciones similares a las que ya hemos comentado de los archivos, como por ejemplo crear y borrar directorios,

mover un directorio a otro, cambiar el dueño de un directorio, etc. El sistema de manejo de archivos por medio del mecanismo de directorios puede realizar la política de compartición de archivos, indicada previamente asignando permisos a los usuarios para leer, escribir y ejecutar archivos que se encuentren en directorios propiedad de otros.

CONTROL DE ACCESO E INTEGRIDAD DE DATOS

La política de proteger al sistema y a los usuarios de la pérdida y corrupción de datos se consigue con los mecanismos de *control de acceso*, que implementan los procesos de *clave de acceso, permisos de archivos* y *modos de archivos*. Para implementar los permisos de archivos los usuarios deben estar clasificados con relación a los archivos que están usando, es decir, acceso propio, acceso de grupo y acceso público. Y dentro de cada uno de estos tipos de acceso, el usuario tiene permiso para usar el archivo en tres formas básicas: lectura, escritura y ejecución. En cada archivo el usuario que lo ha creado fija qué permisos (leer, escribir o ejecutar), se permitirán para cada tipo de acceso. Por ejemplo, en el archivo de un programa que se está desarrollando por un equipo de trabajo, el jefe del equipo puede tener permiso de leer, escribir y ejecutar; el resto de los miembros del equipo pueden tener permiso de lectura y ejecución, y todos los demás (público) sólo permiso de ejecución. Estos modos no son los únicos; otros pueden ser:

Lectura: el usuario sólo puede leer el archivo.
Añadir: el usario sólo puede añadir datos nuevos (normalmente al final del archivo).

Actualizar: el usuario sólo puede reescribir registros ya existentes.
Cambiar protección: el usuario puede cambiar los permisos de acceso.
Borrado: el usuario puede borrar el archivo del sistema.

Los permisos de actualizar, añadir y borrar pueden ser vistos como variantes del permiso de escritura. Si los permisos se clasifican en relación a la importancia de sus efectos, como por ejemplo: borrado, cambio de protección, actualización, añadir, lectura, ejecución y conocimiento, entonces, para un caso determinado, el sistema de archivos puede asumir que se poseen todos los permisos por debajo de uno dado. Por ejemplo, el permiso de borrado presupone que se poseen todos los demás, mientras que el de añadir sólo asume el de lectura, ejecución y conocimiento.

Estos mecanismos sirven para combatir la invasión y la corrupción intencionada, pero el sistema también debe vigilar las pérdidas de integridad debidas a un mal funcionamiento *(hardware* o *software).* Entre los sistemas que se pueden utilizar para prevenir esto, está la obtención incremental de copias de seguridad, que puede ser muy compleja en los sistemas grandes, y la conservación de diversas generaciones de un archivo por si se produjese una corrupción de los datos.

MANEJO E IMPLEMENTACION DE BASES DE DATOS

El crecimiento del uso de grandes archivos en línea accedidos por múltiples programas ha creado algunos problemas en el manejo de archivos. Para solucionar estos problemas se desarrolló un conjunto de técnicas

que al evolucionar crearon los sistemas de base de datos, capaces de manejar gran variedad de tipos distintos de datos. A través del paso del tiempo han surgido diferentes modelos, incluyendo el *relacional*, el *jerárquico* y el de *red*. Debido a la complejidad de estas bases de datos no se hace en este libro un tratamiento completo, ya que eso está fuera del propósito general del libro. Para implementar un sistema de archivos, cada uno de éstos debe estar almacenado físicamente y el sistema operativo debe poseer suficiente información sobre cada archivo para poderlos recuperar y salvar de un modo eficiente. Normalmente se emplea un *definidor del archivo* que se añade a éste como si fuese un bloque separado y contiene información tal como identificación, posición física, organización, descripción del dispositivo, parámetros de control de acceso, tipo de archivo, fecha de creación, etc. La lista varía con la sofisticación del sistema operativo. En el directorio del disco existe una tabla de punteros, cada uno de los cuales señala a una de estas tablas de cada archivo. Cuando se activa un archivo la información de dicho bloque se copia en la memoria principal para ser accedida en un futuro por el sistema operativo cuando se procese el archivo. Al cerrar el archivo la versión actualizada de dicho registro de información se graba en el disco.

Los datos del usuario contenidos en el archivo pueden organizarse de un modo dinámico en el disco; el archivo puede crecer o disminuir en unidades de bloques según haga falta. Cada bloque del disco está en una tabla que indica los bloques que quedan libres. Para evitar la fragmentación del espacio en disco, los bloques se pueden encadenar usando punteros, por lo que es innecesario el uso de áreas contiguas para almacenar grandes archivos. Si todos los punteros se reúnen en un *bloque de índices*, entonces se puede usar esta tabla para incrementar la velocidad de ac -

ceso a un punto dentro de un fichero. Si desarrollamos esta filosofía un paso más, podemos reemplazar el bloque de índices lineal por una *estructura encadenada de punteros;* entonces los borrados e inserciones de bloques de disco también serían eficientes al actualizar un fichero.

ORGANIZACION INDIVIDUAL DE ARCHIVOS

El agrupamiento de los registros en bloques físicos en el disco ofrece una mejora en la eficiencia del almacenamiento; de un modo similar, el sistema en que los registros se organizan dentro del archivo puede tener sus efectos benéficos al acceder y usar el archivo. Si los registros se organizan dentro del archivo en función de la secuencia ordenada de un dato específico del registro (llamado clave), entonces el fichero está organizado *secuencialmente.* Cuando posteriormente se accede a los registros por medio de esta clave, se está realizando un acceso secuencial. Muchas aplicaciones que procesan más del 70 por 100 de los registros de un archivo encontrará que la organización secuencial tiene un acceso muy eficiente, aunque el acceso aleatorio es una alternativa.

La organización *aleatoria* también necesita que haya un campo clave para identificar el registro, pero no requiere que se lean todos los registros de la secuencia entre dos accesos consecutivos. Este efecto se consigue por un programa de control de acceso que traduce el nombre del archivo y el valor de su clave a una posición física del disco, obtenida por medio de una tabla de posiciones en disco existente para dicho archivo.

Un buen ejemplo de aplicaciones que emplean acceso aleatorio son los sistemas de reserva de plazas

en los aviones, diseñados para recuperar y actualizar rápidamente la información de registros aleatorios en tiempo real. Algunas aplicaciones necesitan los beneficios que ofrecen ambos sistemas. Por ejemplo, los recibos del teléfono para enviar al cliente se preparan mejor procesando el archivo en forma secuencial, pero las preguntas telefónicas acerca de los recibos se responden mejor por medio de un acceso aleatorio al mismo fichero. Un tercer método de acceder a un fichero es aquel que permite acceder aleatoriamente a un fichero organizado de forma secuencial (por medio del uso de una tabla de índices), y se llama *archivo indexado secuencial.* En este caso el programa de control debe manejar las inserciones y los borrados, existiendo diversas soluciones comerciales.

Otras organizaciones de archivos son los *invertidos* (que tienen registros con múltiples claves), multilistas y archivos estructurados en árbol, pero por razones de espacio no podemos tratarlos aquí todos. Los programas de control que se encargan de acceder a los archivos se diseñan modularmente para que reflejen los niveles funcionales implicados en el proceso de acceso. Una secuencia típica de manejo de un dispositivo y de un archivo sería: establecer la correspondencia lógica/física del archivo, inicializar las rutinas de acceso, manejar los *buffers* del archivo, determinar las direcciones físicas, controlar las peticiones de entrada/salida, especificar las posiciones en memoria principal y controlar el funcionamiento del dispositivo.

8

Manejo de sistemas

Hasta ahora, para los propósitos de este libro, hemos considerado los recursos del ordenador vistos desde cinco puntos de vista y con la perspectiva de su manejo. Evidentemente, cuando el ordenador está funcionando, todos estos recursos interaccionan dinámicamente y se debe considerar el manejo del sistema como un todo, en lugar de considerar sus partes separadas. Examinado como un todo hay cuatro temas del sistema que se deben estudiar adecuadamente: la *protección* de los recursos; la *contabilidad* del uso del sistema; la *respuesta* general del sistema, y el control del sistema.

Si discutimos la protección de los recursos estamos hablando de cualquier elemento del *hardware* o del *software* que se usa por toda la comunidad de usuarios o por cualquier parte de ella; sin la adecuada protección del sistema de ordenadores se degradará como cualquier otro dispositivo hecho por el hombre. Los aspectos de la contabilidad se aplican evidentemente en cualquier entorno comercial, pero también deben aplicarse en todos los sistemas, incluidos los pequeños de un solo usuario en los que se debe conocer el costo del uso del sistema. Se debe examinar la respuesta general del sistema cuando se espera una

demanda creciente, ya que los datos obtenidos sirven para estimar las demandas futuras y cuándo harán falta mejoras de *hardware* o *software*. El control del sistema se refiere al manejo de tareas por el sistema y puede automatizarse en gran medida por medio de archivos de control de tareas, también denominados archivos EXEC, lo que puede influir en la respuesta general del sistema. En este capítulo examinaremos los sistemas de protección, contabilidad y respuesta del sistema.

PROTECCION DE LOS RECURSOS DEL SISTEMA

La segunda ley de la termodinámica garantiza que cualquier sistema real (o imperfecto) —incluyendo los sistemas de manejo de información— incrementará la entropía; es decir, tenderá a desorganizarse más y a convertirse en más caótico a menos que se realice un esfuerzo externo en el sistema para evitar esta tendencia. Consecuentemente los encargados del sistema, sean hombres o un programa, deben realizar la tarea de controlar la seguridad del sistema dentro de los límites del coste razonable disponible para esta tarea. La pérdida de integridad de los datos debido a accidentes, errores de *hardware* o del sistema operativo o fallos del usuario. El empleo erróneo de la información y el acceso no autorizado, incluso si es inintencionadamente, pueden hacer que la confianza de los usuarios decaiga y, a menos que se introduzca alguna corrección, se producirá un rápido deterioro del servicio. Algunas de las medidas que hay que tomar no son de tipo técnico, sino más bien administrativas o legales. Sin embargo, las medidas más importantes que se deben tratar son las resultantes del acceso no autorizado al sistema y la

incapacidad del sistema de restringir el acceso de los usuarios autorizados a ciertos recursos. Los mecanismos de protección que se encuentran más frecuentemente son:

Identificación: Que permite reconocer al usuario habitual mediante una clave de acceso *(password)* introducida en el terminal; el cifrado de la clave por sistemas no reversibles asegura esta protección incluso si se conoce el algoritmo de cifrado y la versión cifrada.

Claves de acceso de una sola vez: El sistema puede emplear una clave de este tipo con cada usuario, pidiéndole cada vez la siguiente palabra de una lista.

Terminales conectados por modem:.: Estos son más difíciles de proteger, ya que se puede intervenir la línea o controlarla por radiofrecuencia.

Restricciones: Se debe restringir a los usuarios autorizados el uso de determinadas rutas de acceso y recursos para prevenir la destrucción accidental o intencionada o la corrupción de datos o servicios.

Las entidades a las que se puede acceder se llaman objetos (archivos, páginas, procesos o dispositivos) y las que realizan el acceso se llaman sujetos (normalmente procesos del sistema o de un usuario). Un sujeto ejercita un derecho de acceso (leer, escribir, ejecutar) y puede transferir dicho derecho a un objeto que a su vez (como los sujetos) llame a otro objeto. El programa de control de accesos utiliza una *matriz de derechos de acceso* que almacena los derechos de acceso de cada par de sujeto-objeto, y se emplea dicha matriz cada vez que se solicita un acceso. Se ha empleado esta técnica y otras similares para proporcionar protección a los recursos y son necesariamente complicadas, dado que proteger correc -

tamente todos los recursos compartidos de un sistema es muy complejo.

CONTABILIDAD

Los aspectos económicos del manejo de un ordenador pueden ser considerables en los grandes sistemas y, si no se les maneja correctamente, pueden contribuir como cualquier mal funcionamiento técnico del sistema a la caída de éste. El coste total del funcionamiento del sistema debe compartirse por los usuarios o por su organización. En un entorno comercial la cantidad que debe pagar cada usuario debe deducirse del uso que realiza cada cuenta. Esto sólo puede evitarse si el ordenador ha sido comprado por una organización que ofrece el sistema como un servicio a sus miembros. En caso contrario hay que realizar una *contabilidad de recursos* y llevar una *política de precios* para poder facturar individualmente.

Contabilidad de recursos: Un considerable número de recursos puede estar sujeto a una medida de su uso, aunque no todos los sistemas contarán con los servicios necesarios de control. Algunas de las cantidades que se miden son: tiempo empleado total; tiempo de CPU; tiempo de conexión (del terminal del usuario al sistema); tiempo empleado en acceder al disco; espacio ocupado en disco; espacio ocupado en memoria principal; tiempo de ocupación (por ejemplo, 10 páginas de memoria ocupadas durante 3,25 segundos dan 32,5 páginas-segundos); tiempo de transmisión de entrada y salida, y segundos empleados en lecturas y escrituras.

Política de precios: Las medidas anteriores están sujetas a una política de precios que, idealmente, debería ser sencilla, reproducible e imparcial. Al ser reproducible se asegura que el usuario puede calcular el costo del servicio con las estadísticas de uso que se proporcionan. La existencia de un sistema de contabilidad en línea permite calcular de forma inmediata un balance, de modo que antes de empezar una tarea se puede ver el límite de crédito de dicha cuenta.

Los sistemas grandes y complejos usan técnicas sofisticadas de contabilidad de recursos y sistemas igualmente complejos de calcular los precios de las cuentas individuales. Aunque esto no haga falta, la contabilidad de recursos sigue siendo un requerimiento para determinar qué recursos necesitan desarrollarse para mejorar el nivel de servicios ofrecidos a los usuarios.

MEDIDA DE LA RESPUESTA

El manejo de los sistemas también se extiende a la mejora del funcionamiento de la instalación. La máquina y el *software* pueden reconfigurarse o sintonizarse según la demanda creciente indique, para obtener una respuesta adecuada si no es la mejor posible. Esta respuesta se mide de diversos modos utilizando diversos dispositivos *hardware* y *software,* entre éstos se incluyen:

Velocidad de respuesta: Que mide el retraso que se procede entre la petición de un servicio y la recepción de la respuesta por el sistema. Esta medida es importante para los usuarios individuales.

Disponibilidad: Mide la probabilidad de que el servicio esté disponible en un momento dado; esto, desde luego, dependerá de la hora del día, ya que en determinados momentos el sistema estará sobrecargado, mientras que en otros estará vacío.

Utilización: Se aplica a recursos específicos como la CPU o las unidades de disco y mide el porcentaje de tiempo que dicho dispositivo es usado.

Estas medidas se pueden realizar por medio de monitores *hardware /software,* pero ambos necesitan interactuar hasta cierto punto con el sistema que se está monitorizando. Los monitores por *hardware* imponen menos cargas al sistema que los monitores de *software,* pero ambos son capaces de adquirir grandes cantidades de datos empíricos que deben tratarse antes de poder interpretarlos. Basándose en las diversas medidas realizadas se pueden obtener predicciones sobre las futuras cargas del sistema o cómo debe reconfigurarse. Para comprobar el funcionamiento del sistema con los cambios recomendados se emplea un sistema de modelos antes de implementar los cambios. Cuando éstos son pequeños y reversibles, se utiliza el sistema mismo con el sistema de prueba y error. Cuando los cambios a realizar son más drásticos hay que emplear un modelo de simulación o hacer un estudio analítico. Los modelos analíticos usan un conjunto de ecuaciones que expresan la relación aproximada entre las medidas en uso. Los modelos de simulación usan el sistema para proporcionar medidas en base a un trabajo abstracto, midiendo la respuesta del sistema por medio de interpolaciones o extrapolaciones de los resultados. Un test de velocidad *(benchmark)* es un trabajo cuya ejecución se considera típica del trabajo que se debe ejecutar realmente.

9

Algunos sistemas operativos para microordenadores

Sólo recientemente han empezado a surgir diferencias entre los sistemas operativos y sus características. A finales de 1970 sólo había un competidor real en la generación de microordenadores de 8 bits: el CP/M. Hoy en día existen más de 80 sistemas operativos para máquinas de 16 bits y el usuario, ya sea profesional o *hobbista*, no sólo tiene que buscar los programas que necesite, sino también el sistema operativo que más se adecúe a sus necesidades, ya que la potencia de éstos sistemas es tan elevada que hace falta un sistema operativo muy sofisticado que sea capaz de manejarlos. Actualmente existe un mínimo de 10 sistemas operativos y en la mayoría de los casos estos están disponibles en un amplio rango de máquinas.

Tres de estos sistemas se originaron en máquinas de 8 bits (CP/M, Apple DOS y UCSD P-system), pero siguen teniendo actualidad, sobre todo el P-system, que ofrece portabilidad a las casas de *software* y resulta eficiente emulando a las máquinas de 16 bits.

La batalla principal del mercado en 16 bits se desarrolla hoy en día entre los sistemas operativos del estilo del CP/M (MS-DOS, PC-DOS y CP/M 86) y los que provienen realmente de entornos de miniordenadores (como el UNIX y el PICK). La función del sistema operativo consiste en actuar como intermediario eficaz entre los programas de aplicación del usuario y el *hardware*. Un componente muy importante en este cometido es el manejo de los discos y el sistema de archivos que lo controla. Con una cantidad limitada de memoria principal es necesario llegar a un acuerdo sobre la cantidad de memoria que se destina al núcleo para agilizar la respuesta del sistema y la que se destina a los programas de usuario y a los *buffers* de datos. Cada sistema operativo emplea una política diferente en función del mercado al que vaya destinado. Por esta razón puede resultar interesante examinar los productos existentes en el mercado.

Este capítulo examina muy brevemente los sistemas operativos más importantes existentes para microordenadores. Su popularidad se debe a que intentan aportar los requerimientos esenciales necesarios para manejar los recursos del *hardware*. Es decir: asociación flexible de los dispositivos lógicos a los físicos; ejecutar y controlar las operaciones de entrada/salida; organizar la carga de programas y la comunicación entre programas; procesar interactivamente las órdenes del usuario; actualizar los directorios en disco; realizar las manipulaciones de archivos; llevar control de la contabilidad de los usuarios; proteger los datos, y procesar las interrupciones de *hardware*.

UNIX

UNIX es el nombre que recibe un sistema operativo desarrollado en 1970 por Ken Thompson, un ingeniero de informática que trabajaba en los laboratorios Bell. Antes de convertirse en un producto comercial este sistema operativo ha sido utilizado durante varios años en centros de investigación y universidades. Está soportado por un considerable número de microordenadores de 16 bits orientados al mercado comercial y el número de instalaciones que usan UNIX ha aumentado en un 500 por 100 hasta alcanzar 100.000 en 1983 y probablemente alcance los 400.000 en 1986.

El UNIX ofrece muchas características interesantes para realizar desarrollos de propósito general y en aplicaciones informáticas. Entre estas características cabe destacar: Una estructura jerárquica de ficheros que acepta volúmenes montables; entradas/salidas compatibles entre archivos, dispositivos y entre procesos; multiproceso y manejo de múltiples usuarios; diversos intérpretes de órdenes en función de los requerimientos del usuario, y un gran rango de utilidades, funciones del sistema y lenguajes. El UNIX mantiene múltiples ficheros y directorios para su uso. Todos los directorios parten de un directorio raíz creado por el sistema. Cada directorio inferior puede contener archivos del usuario (textos o programas), directorios o archivos especiales (usadas para interconectarse con otros dispositivos). Cuando se crea un nuevo usuario automáticamente se le asigna un directorio base a partir del cual se pueden desarrollar nuevos directorios.

El UNIX emplea las definiciones estándar de entrada/salida y está construido para aceptar y enviar datos por medio de estas definiciones. Esta capacidad

hace que la redirección de entrada/salida sea muy fácil y le otorga una gran potencia al encargado del sistema. Las "tuberías" aplican el mismo concepto al flujo de datos entre programas: se pueden combinar varios programas para que se pasen los datos entre unos y otros y realicen un proceso general.

Considerando su tamaño y capacidades, el UNIX es muy portable debido a que está escrito en un lenguaje de alto nivel que permite su rápida implementación por las casas de *software.* Microsoft ya soporta el UNIX con una versión propia denominada XENIX y Digital Research ha decidido soportar aplicaciones realizadas en UNIX. No obstante, la documentación todavía refleja sus orígenes académicos, punto que espera verse resuelto pronto con los libros que publiquen editoriales independientes. Debido a la generosidad de los Bell Laboratories en la concesión de licencias a las universidades, gran cantidad de los ingenieros de reciente graduación lo saben manejar, lo que constituye un fuerte incentivo para la implementación de este sistema.

PICK

Los nuevos microprocesadores de 16 y 32 bits permiten la realización de familias de ordenadores compatibles que sólo se diferencian en costo y potencia. Estos desarrollos implican el diseño de un nuevo grupo de sistemas operativos que no sean dependientes de una máquina en particular. El UNIX y el UCSD P-system se orientan hacia este objetivo. Por otro lado el PICK (que recibe el nombre de Richard Pick) ofrece la misma flexibilidad y potencia de uso que el UNIX, pero con un interfaz de usuario mucho más sencillo. Sus creadores han resuelto el problema de la comunicación entre los programas de

aplicación y el sistema operativo haciendo que éste ofrezca un amplio rango de utilidades centradas en una base de datos relacional. El usuario realiza las llamadas a las funciones de un modo no basado en procedimientos (es decir, sin usar un programa). Si se usa un lenguaje no basado en procedimientos sólo hace falta indicarle al ordenador lo que se desea hacer; el *software* del sistema examinará cómo se realiza esto y ejecuta las instrucciones del sistema operativo necesarias para conseguir esto. Por ejemplo, con el lenguaje del PICK se puede decir: "LIST ALL EMPLEADOS WITH SUELDO>80000" (lista todos los empleados con sueldo superior a 80000); esta frase haría que se activasen todas las utilidades necesarias para proporcionar la información.

Todos los datos dentro del PICK se almacenan empleando un algoritmo especial, de modo que se pueden recuperar haciendo un solo acceso a disco y cada elemento de datos es de longitud variable. Los archivos se organizan de forma relacional; cada uno tiene asociado un diccionario de datos en el que se almacenan todos los parámetros de definición de campos y atributos; cada usuario también tiene un diccionario maestro de todos los archivos y, a su vez, el sistema guarda un diccionario de todos los diccionarios maestros. Otra característica avanzada del sistema es que se puede permitir el acceso a datos almacenados en diversas cuentas. El PICK ofrece un gran número de funciones de alto nivel: editor de líneas (EDITOR), compilador (DATA BASIC), composición electrónica (RUNOFF), proceso de textos (JET), utilidades de impresión (SPOOLER), proceso de procedimientos (PROC) y proceso de instrucciones de terminal (TCL). Se espera que aparezca una nueva versión que ofrezca lenguaje C y permita al UNIX funcionar bajo control del PICK. Las características atractivas del PICK implican que éste se creará su

propio mercado y tendrá sus propios proveedores, en especial casas de *software* que quieren un camino rápido para proporcionar *software* fácil de manejar a usuarios no especializados.

BOS, OASIS Y OTROS SISTEMAS OPERATIVOS INTEGRADOS

El sistema operativo BOS (Business Operating Systems) es un intento de los usuarios de COBOL de hacer que sus programas sean transportables a una gran variedad de máquinas por medio del uso de un P-system (P, por pseudo-máquina). Es altamente portable y multiusuario y tiene una buena reputación por su calidad, pero su base de programas está algo limitada debido al dialecto propio de COBOL que emplea.

Otro competidor en el mercado de los sistemas multiusuarios de gestión es el OASIS. Este sistema operativo está disponible en versiones de 8 y 16 bits. Sus defensores indican que es tan potente como el UNIX pero tan fácil de usar como el PICK y permite desarrollar aplicaciones en COBOL o BASIC. Es un sistema operativo seguro y potente, pero tiene menos programas que sus rivales, lo que podría hacer que perdiera el tren.

Los sistemas operativos integrados, de los que ha sido pionero el Smalltalk, representan una revolución en la forma que emplea el usuario para comunicarse con la máquina. Con estos sistemas tienden a desaparecer las diferencias existentes entre el sistema operativo y los programas de aplicaciones. La concurrencia que tienen algunos de ellos permite ejecutar varias aplicaciones simultáneamente, mostrándose cada una en una ventana y realizándose el control por

medio de un ratón que controla a iconos dibujados en pantalla. Actualmente existen varios productos que usan sistemas operativos integrados: los Lisa y Macintosh de Apple, ICL y el portable de Gavilan. Estos productos están orientados al usuario individual que no necesita complicados sistemas operativos multitarea. Como inconveniente se encuentra la mayor dificultad que representa hacer programas basados en las mismas directrices.

CP/M

El CP/M (Control Program for Microcomputers, Programa de Control para Microordenadores) es un sistema operativo para microordenadores desarrollado por Digital Research y empleado por un gran número de fabricantes en ordenadores que incorporen los microprocesadores 8080, 8085 ó Z80 y que empleen un sistema de discos. Desde 1975 hasta la actualidad se han vendido más de 500.000 sistemas que emplean este sistema operativo y se han desarrollado de 2.000 a 3.000 programas distintos. Por estas razones el CP/M se ha convertido en un estándar de la industria; sin embargo, esto no indica que sea perfecto, ya que la mayoría de los usuarios opinan que no es fácil de usar y propenso a generar errores del usuario. Debido a esto existe un amplio mercado de libros explicando todo lo referente al CP/M. Un inconveniente importante de este sistema es el gran número de formatos de disco que soporta, que implica que se debe crear una tabla especial, denominada CBIOS, y que puede hacer que los datos que se creen en máquinas diferentes sean incompatibles.

El CP/M se carga en cuatro áreas separadas de la memoria: El área de arranque (BOOT), el sistema básico de entrada/salida (BIOS), el sistema básico

de discos (BDOS), el procesador de instrucciones de consola (CCP) y el área de programas transitorios (TPA). La cantidad de memoria que utiliza cada área se determina por el total disponible. En el núcleo del CP/M se incluyen las funciones *intrínsecas* que están disponibles al usuario por medio del CCP. Todas las demás órdenes están almacenadas en disco como programas transitorios (o archivos de órdenes) que se leen en la TPA para ejecutarse.

Al igual que el Apple DOS, el CP/M sólo ofrece el mínimo necesario para que el sistema sea operacional. Las funciones básicas disponibles incluyen manejo de dispositivos (STAT), manejo de archivos (REN, ERA, TYPE, DIR), manipulación de archivos (PIP, LOAD, DUMP, SAVE) y manejo del sistema (USER, MOVCPM, FORMAT, SYSGEN). De todas éstas, sólo DIR, TYPE, SAVE y REN están incluidas en el núcleo y el resto se almacenan en el disco como programas transitorios. Todas las aplicaciones se presentan bajo el formato de programas transitorios.

Pero el CP/M no se ha quedado parado y evoluciona con los tiempos. Recientemente se ha introducido una versión denominada CP/M 86, destinada a competir con el MS-DOS, que permite concurrencia y multitarea, y el anuncio de un *chip* que incorpora el CP/M puede devolverle algo de popularidad a este sistema operativo.

MS-DOS

El MS-DOS es la versión mejorada del sistema operativo de 16 bits de Microsoft, y representa un ejemplo típico de la tercera generación de sistemas operativos para microordenadores basados en el 8086 u 8088 de Intel. Normalmente estos ordenadores de 16 bits tienen 128 Kbytes de memoria principal y uno

o dos discos, lo que permite la existencia de sistemas operativos más sofisticados. La versión 2.0 y posteriores de este sistema operativo tiene varias características útiles, como son *los manejadores instalables de dispositivos (Device Drivers)*, que son rutinas encargadas de controlar a dispositivos periféricos tales como el teclado, el monitor, los discos, la impresora, etcétera. De este modo los fabricantes independientes de dispositivos pueden interconectar sus aparatos más fácilmente a ordenadores que incorporen el MS-DOS. Este sistema operativo también es compatible con el XENIX en un gran número de áreas: manejo de archivos, modo ejecutable (modo de órdenes), órdenes del sistema (transportables al XENIX) y una estructura de archivos y directorios similar en ambos. Otras características típicas de XENIX que también ha adoptado el MS-DOS son la redirección de entrada/salida y la comunicación entre procesos por medio de tuberías, que permiten pasar la salida de un programa directamente como entrada de otro. El MS-DOS se ha diseñado para el mercado internacional en mente y ofrece la posibilidad de trabajar con varios juegos de caracteres y con ciertos formatos en función del país. Esta característica es similar a la existente en los *mainframes* de IBM. Como ejemplo está el uso de la coma o punto decimal, dependiendo del país. Otra posibilidad de esta internacionalización es que los fabricantes de productos pueden hacer que éstos den los errores en los idiomas correspondientes a cada país. Finalmente, y aunque el MS-DOS no es realmente multitarea, se le puede hacer trabajar de modo que ejecute dos programas a la vez. Un programa que hace buen uso de esta capacidad es el *spooler* de impresora.

El entorno de ventanas de Microsoft,*Windows*, ha sido adoptado por un gran número de fabricantes de ordenadores (23), que intentan evitar los inconvenientes del CP/M estándar de 8 bits. Por tanto, será

posible dentro de poco ejecutar los mismos programas en máquinas distintas y con varios programas ejecutándose a la vez.

UCSD P-SYSTEM

Mucha gente opina que este sistema operativo debería establecerse como el estándar para máquinas de 16 bits y superiores. Se desarrolló en la Universidad de California, en San Diego, y está basado en un sistema capaz de ejecutar el código P producido por su compilador de PASCAL. Este pseudocódigo está ideado para ejecutarse en una máquina virtual, denominada máquina P, y que es el centro de la compatibilidad del sistema. En efecto, todo programa producido por este sistema está pensado para ejecutarse en la máquina P, y dado que ésta es una emulación, puede ejecutarse en cualquier ordenador independientemente del procesador que lleve. En otras palabras, se obliga a cada procesador a emular la misma máquina P.

Esta soporta la mayoría de las características de los microprocesadores estándar, como es el direccionamiento variable, matrices de bytes, campos empaquetados, variables dinámicas, procedimientos solapables y otras características típicas, además de un sistema básico de entrada y salida. Debido a que el P-system es una emulación, sus creadores pudieron idear el diseño ideal. Eligieron palabras de 16 bits, cuando 8 bits era lo estándar, usando dos bytes contiguos. La emulación se realiza al ejecutar el programa, por lo que si el microprocesador es de 8 bits, el sistema es más lento que con otros sistemas operativos. No obstante, con la llegada de los microprocesadores de 16 bits la diferencia de velocidad no es tan significativa y las ventajas que aporta la

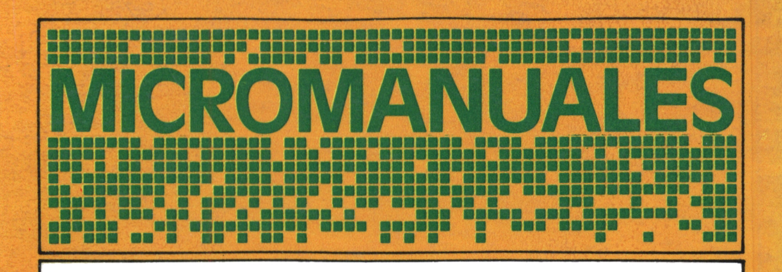
portabilidad pueden ser importantes a la hora de desarrollar programas, ya que el éxito de un sistema depende del número de programas existentes. Y el número de programas depende del número de clientes que deseen comprar una copia. Debido a la portabilidad del P-system la base de ordenadores y, por tanto, el número de usuarios puede incrementarse con respecto a los sistemas operativos no portables.

APPLE DOS

El sistema operativo de Apple, Apple DOS, empezó a estar disponible en 1978 para soportar los ordenadores Apple II basados en disco. Esta máquina de 8 bits ha tenido un éxito increíble, siendo líder mundial en ventas y base de *software* durante muchos años y sólo recientemente ha sido desplazado por el MS-DOS y el CP/M. Su diseño implementa la política de aumentar el área de memoria destinada al usuario reduciendo al mínimo la cantidad necesaria para el DOS. La versión actual es DOS 3.3 y no se espera que se produzca una nueva, ya que Apple ha sacado un nuevo sistema operativo llamado PRO-DOS destinado a las casas de *software* que necesitan utilidades más avanzadas para el desarrollo de programas. El Apple II tiene un núcleo que maneja entradas/salidas de caracteres, salidas por altavoz, entradas de botón y *paddle,* entrada/salida de cassette y un procedimiento de arranque que, si al encender la máquina está el disco conectado, instala el sistema operativo. Entre las utilidades de éste se incluyen funciones para inicializar discos, leer y salvar programas, listar el directorio de un disco y renombrar y borrar los ficheros del disco. Este sistema fue adecuado para miles de casas de *software* que durante un período de seis años han creado programas para

cualquier situación concebible. Es esta gran base de programas la que asegura que el Apple II seguirá viviendo durante mucho tiempo, pese a la competencia de las máquinas de 16 bits. Como punto final hay que indicar que existe un microprocesador de 16 bits compatible con el 6502 que utiliza Apple, lo que parece asegurar el futuro de esta máquina.

Este examen de sistemas operativos para microordenadores ha intentado señalar las ventajas e inconvenientes de cada uno. El usuario que busque un ordenador debe buscar primero un sistema operativo en el cual funcionen las aplicaciones que necesita. Aunque este último punto tienda a desaparecer con la aparición de sistemas operativos que cada vez integran más funciones.



Los MICROMANUALES son libros de referencia condensados que le ofrecen información práctica sobre todas las áreas de la microinformática: programación, aplicaciones, uso de programas, proyectos, etc.

Otros títulos de la colección:

- dBASE III
- Lenguaje ensamblador 8088/8086
- Proyectos de música con microordenadores
- SYMPHONY
- Introducción a las comunicaciones



Sistemas Operativos

Blackburn & Taylor


AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
ACME
https://acpc.me/